C 8

```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
```

FILEID**XMDRIVER

```
XX      XX  MM      MM  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  MM      MM  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  MMMM  MMMM  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
XX      XX  MMMM  MMMM  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    MM  MM  MM  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    MM  MM  MM  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    XX      MM      MM  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    XX      MM      MM  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
  XX  XX    MM      MM  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
  XX  XX    MM      MM  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
XX      XX  MM      MM  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  MM      MM  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  MM      MM  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
XX      XX  MM      MM  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  XMDRIVER - VAX/VMS DMC11/DMR11 Device Driver
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;++
0000    27 ; FACILITY:
0000    28 ;
0000    29 ;       VAX/VMS DMC11/DMR11 Device driver
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ;       This module contains the DMC11/DMR11 driver FDT routines,
0000    34 ;       interrupt dispatcher, interrupt service and fork routines.
0000    35 ;
0000    36 ; AUTHOR:
0000    37 ;
0000    38 ;       R.HEINEN 24-AUG-77
0000    39 ;
0000    40 ; MODIFICATION HISTORY:
0000    41 ;
0000    42 ;       V03-023 RNG0023         Rod N. Gamache          17-May-1984
0000    43 ;               Set the DEV$M_AVL bit to make XM units available.
0000    44 ;
0000    45 ;       V03-022 RNG0022         Rod N. Gamache          29-Feb-1984
0000    46 ;               Fix problem with allocation of map registers which causes
0000    47 ;               too many map registers to be allocated.
0000    48 ;
0000    49 ;       V03-021 RNG0021         Rod N. Gamache          29-Oct-1983
0000    50 ;               Fix broken register useage caused by use of TIMEDWAIT macro.
0000    51 ;
0000    52 ;       V03-020 RNG0020         Rod N. Gamache          27-Jul-1983
0000    53 ;               Changed WAIT10 macro to use system TIMEDWAIT macro.
0000    54 ;               Change all NOP wait loops to use TIMEDWAIT macro.
0000    55 ;               Don't do BUG_CHECK if input request was processed by
0000    56 ;               Interrupt Service Routine.
0000    57 ;
```

```
0000     58 ;        V03-019 ROW0169         Ralph O. Weber           3-MAR-1983
0000     59 ;                Add $IPLDEF.
0000     60 ;
0000     61 ;        V03-018 RNG0012         Rod Gamache     28-Jan-1983
0000     62 ;                Add code to hang up modems on LINE DOWN requests.
0000     63 ;
0000     64 ;        V03-017 RNG0011         Rod Gamache     17-Dec-1982
0000     65 ;                Speed up the startup time for devices that don't run
0000     66 ;                micro-diagnostics.
0000     67 ;
0000     68 ;        V03-016 RNG0010         Rod Gamache     04-Nov-1982
0000     69 ;                Setup timeout routine offset for new fork process
0000     70 ;                added to transmit process routine.
0000     71 ;
0000     72 ;        V03-015 RNG0009         Rod Gamache     07-Sep-1982
0000     73 ;                Fix cancel routine to only abort user's I/O on $CANCEL
0000     74 ;                request and not to shutdown device or abort other users'
0000     75 ;                I/O. Add another fork block to UCB to allow a fork on tranmit
0000     76 ;                requests - allows users to transmit any size message up to
0000     77 ;                16K. Remove the code to pre-allocate one transmit map register.
0000     78 ;
0000     79 ;        V03-013 RNG0008         Rod Gamache     01-Sep-1982
0000     80 ;                Reduce startup time required in previous enhancement.
0000     81 ;                Fix startup problem when running in LOOPBACK mode - fixes
0000     82 ;                problem found by UETP.
0000     83 ;
0000     84 ;        V03-012 RAN0001         R. Newell       08-Jul-1982
0000     85 ;                Add code to determine whether a DMC has the high-speed or
0000     86 ;                low-speed microcode chip set and what the mode and interface
0000     87 ;                switches are set to on a DMR.
0000     88 ;
0000     89 ;
0000     90 ; PREVIOUS MODIFICATIONS:
0000     91 ;
0000     92 ;                        Al Eldridge, Scott Davis, Len Kawell, Rod Gamache   1979-1982
0000     93 ;--
```

```
0000     95 ;
0000     96 ; System definitions
0000     97 ;
0000     98         $ACBDEF                         ; AST control block
0000     99         $CANDEF                         ; Define Cancel reason codes
0000    100         $CRBDEF                         ; Controller request block
0000    101         $CXBDEF                         ; Define CXB block
0000    102         $DCDEF                          ; Device types
0000    103         $DDBDEF                         ; Device data block
0000    104         $DPTDEF                         ; Driver prologue table
0000    105         $DYNDEF                         ; Dynamic data structure types
0000    106         $FKBDEF                         ; Fork block definitions
0000    107         $IDBDEF                         ; Interrupt data block
0000    108         $IODEF                          ; I/O functions
0000    109         $IPLDEF                         ; IPL symbolic definitions
0000    110         $IRPDEF                         ; I/O packets
0000    111         $JIBDEF                         ; Job information block
0000    112         $NMADEF                         ; Network management codes
0000    113         $PCBDEF                         ; Process control block
0000    114         $PRDEF                          ; Processor registers
0000    115         $SSDEF                          ; System status codes
0000    116         $TQEDEF                         ; Timer Queue Element
0000    117         $UBADEF                         ; UNIBUS adapter registers
0000    118         $UCBDEF                         ; Unit control block
0000    119         $VADEF                          ; Virtual address fields
0000    120         $VECDEF                         ; Interrupt vector
0000    121         $XMDEF                          ; XMDRIVER symbols
0000    122
0000    123 ;
0000    124 ; Local macros
0000    125 ;
0000    126         .MACRO  SETBIT  POS,BAS,?L      ; Set a single bit
0000    127                 BBSS    POS,BAS,L
0000    128         L:
0000    129         .ENDM   SETBIT
0000    130
0000    131         ;*******
0000    132
0000    133         .MACRO  CLRBIT  POS,BAS,?L      ; Clear a single bit
0000    134                 BBCC    POS,BAS,L
0000    135         L:
0000    136         .ENDM   CLRBIT
0000    137
0000    138         ;*******
0000    139
0000    140         .MACRO  ADDLC   COUNT,COUNTER,?L; Add to counter
0000    141                 ADDL    COUNT,COUNTER   ; Increment
0000    142                 BCC     L               ; Br if no carry
0000    143                 MNEGL   #1,COUNTER      ; Set to maximum value
0000    144         L:
0000    145         .ENDM   ADDLC
0000    146
0000    147         ;*******
0000    148
0000    149         .MACRO  WAIT10 WTIME,?L1
0000    150
0000    151         TIMEDWAIT       TIME=WTIME,-
```

```
                                                        INS1=<BITB      S^#0,S^#0>,-
                                                        INS2=<BNEQ      L1>,-
                                                        DONELBL=L1

                        .ENDM   WAIT10

                        ;*******

                        .MACRO  COUNTER TYPE,BITMAP=NO,WIDTH=8,-
                                        BASEOFF1=0,UCBOFF1=DEVCNT,BASEOFF2=0,UCBOFF2=DEVCNT
                        $$$TYP = NMA$C_CTCIR_'TYPE' & NMA$M_CNT_TYP
                        .IIF IDN <BITMAP><YES>, $$$TYP = $$$TYP!<NMA$M_CNT_MAP>
                        $$$WID = 0                      ; Set reserved mask field
                        .IIF IDN <WIDTH><8>, $$$WID = <1@NMA$V_CNT_WID>
                        .IIF IDN <WIDTH><16>, $$$WID = <2@NMA$V_CNT_WID>
                        .IIF IDN <WIDTH><32>, $$$WID = <3@NMA$V_CNT_WID>
                        .IIF EQ $$$WID, .ERROR          ; Invalid bit width value
                        .WORD   NMA$M_CNT_COU!$$$WID!$$$TYP
                        .IF  NE BASEOFF1
                                        .BYTE BASEOFF1,UCB$B_XM_'UCBOFF1'-UCB$B_XM_DEVCNT
                        .IIF NE BASEOFF2, .BYTE BASEOFF2,UCB$B_XM_'UCBOFF2'-UCB$B_XM_DEVCNT
                                        .BYTE 0
                        .ENDC
                        CNT_BUFSIZ = CNT_BUFSIZ + 2 + <WIDTH/8>
                        .IIF IDN <BITMAP><YES>, CNT_BUFSIZ = CNT_BUFSIZ + 2
                        .ENDM   COUNTER

                ;
                ; Local symbol definitions
                ;

                ;
                ; $QIO parameter offsets
                ;
00000000        P1      = 0                             ; Parameter 1
00000004        P2      = 4                             ; Parameter 2
00000008        P3      = 8                             ; Parameter 3

00000100        BASETAB_SIZE    = 256                   ; Size of base table
00003FFF        MAX_C_BUFSIZE   = 16383                 ; Maximum transfer size
00000007        MAX_RCV         = 7                     ; Maximum number outstanding receives
00000007        MAX_XMT         = 7                     ; Maximum number outstanding transmits
00000003        DMC_DMR         = 3                     ; DMC or DMR test value
000F4240        SHUT_TIME       = 1000*1000             ; Shutdown delay time (100 ms)
00002296        UINST_CNF       = ^O21226               ; Microinstruction to get config
0000814D        UINST_RROM      = ^O100515              ; Microinstruction to read DMC ROM
00000390        LS_UCODE        = ^O1620                ; Contents of addr 0115 in l.s. u-code
0000A40B        DROP_DTR        = ^O122013              ; Drop DTR on modem
00000082        EXECUTE_UC      = ^O202                 ; Execute in DMC PORT

                ;
                ; XMDRIVER UCB extensions
                ;
                        $DEFINI
00000090        . = UCB$C_LENGTH

                $DEF    UCB$Q_XM_QUEUES                 ; Message and I/O request queue heads
                $DEF    UCB$Q_XM_XMT_REQ .BLKQ  1       ; Transmit I/O requests awaiting start
```

```
                     0098   209 $DEF     UCB$Q_XM_RCV_REQ .BLKQ  1        ; Receive I/O requests awaiting message
                     00A0   210 $DEF     UCB$Q_XM_PORT    .BLKQ  1        ; Transmits/receives awaiting the port
                     00A8   211 $DEF     UCB$Q_XM_XMT_PND .BLKQ  1        ; Transmit I/Os given to device
                     00B0   212 $DEF     UCB$Q_XM_RCV_PND .BLKQ  1        ; Receive buffers given to device
                     00B8   213 $DEF     UCB$Q_XM_POST    .BLKQ  1        ; Transmits/receives awaiting posting
                     00C0   214 $DEF     UCB$Q_XM_RCV_BUF .BLKQ  1        ; Free receive buffers
                     00C8   215 $DEF     UCB$Q_XM_RCV_MSG .BLKQ  1        ; Receive buffers containing messages
00000008             00D0   216 UCB$C_XM_QUEUES = <.-UCB$Q_XM_QUEUES>/8 ; Number of queue heads
                     00D0   217
                     00D0   218 $DEF     UCB$L_XM_RCV_MAP .BLKL  MAX_RCV  ; Receive mapping vector
                     00EC   219 $DEF     UCB$L_XM_XMT_MAP .BLKL  MAX_XMT  ; Transmit mapping vector
                     0108   220 $DEF     UCB$B_XM_RCV_MAP .BLKB  1        ; Receive mapping in use flags
                     0109   221 $DEF     UCB$B_XM_XMT_MAP .BLKB  1        ; Transmit mapping in use flags
                     010A   222 $DEF     UCB$B_XM_RCV_MAX .BLKB  1        ; Maximum concurrent receives
                     010B   223 $DEF     UCB$B_XM_XMT_MAX .BLKB  1        ; Maximum concurrent transmits
                     010C   224
                     010C   225 $DEF     UCB$W_XM_QUOTA   .BLKW  1        ; Starter's byte quota deducted
00000110             010E   226                      .BLKW  1        ; (spare for alignment)
                     0110   227 $DEF     UCB$L_XM_PID     .BLKL  1        ; Starter's process ID
                     0114   228 $DEF     UCB$L_XM_AST     .BLKL  1        ; Attention AST list
                     0118   229 $DEF     UCB$L_XM_BASETAB .BLKL  1        ; Base table address
                     011C   230 $DEF     UCB$L_XM_BASEMAP .BLKL  1        ; Base table map regiser number/count
                     0120   231
                     0120   232 $DEF     UCB$L_XM_DRVCNT                  ; Driver counters
                     0120   233 $DEF     UCB$L_RCVBYTCNT .BLKL   1        ; Receive byte count
                     0124   234 $DEF     UCB$L_XMTBYTCNT .BLKL   1        ; Transmit byte count
                     0128   235 $DEF     UCB$L_RCVMSGCNT .BLKL   1        ; Receive message count
                     012C   236 $DEF     UCB$L_XMTMSGCNT .BLKL   1        ; Transmit message count
00000004             0130   237 UCB$C_XM_DRVCNT = <.-UCB$L_XM_DRVCNT>/4
                     0130   238
                     0130   239 $DEF     UCB$B_XM_DEVCNT                  ; Device counters
                     0130   240 $DEF     UCB$B_XM_NBFR    .BLKB  1        ; NAKs rcvd - no buffer (DMR11)
                     0131   241 $DEF     UCB$B_XM_HCER    .BLKB  1        ; NAKs rcvd - header BCC error (DMR11)
                     0132   242 $DEF     UCB$B_XM_DCER    .BLKB  1        ; NAKs rcvd - data BCC error
                     0133   243 $DEF     UCB$B_XM_NBFS    .BLKB  1        ; NAKs sent - no buffer
                     0134   244 $DEF     UCB$B_XM_HCES    .BLKB  1        ; NAKs sent - header BCC error
                     0135   245 $DEF     UCB$B_XM_DCES    .BLKB  1        ; NAKs sent - data BCC error
                     0136   246 $DEF     UCB$B_XM_REPS    .BLKB  1        ; REPs sent
                     0137   247 $DEF     UCB$B_XM_REPR    .BLKB  1        ; REPs rcvd
00000008             0138   248 UCB$C_XM_DEVCNT = .-UCB$B_XM_DEVCNT
                     0138   249
                     0138   250 $DEF     UCB$B_XM_FKB     .BLKB  FKB$C_LENGTH ; Fork process fork block
                     0150   251 $DEF     UCB$W_XM_MODSIG  .BLKW  1        ; Modem signals
                     0152   252 $DEF     UCB$C_XM_LENGTH                  ; Size of XMDRIVER UCB
                     0152   253
00000148             0152   254 . = UCB$B_XM_FKB+FKB$L_FR3
                     0148   255
                     0148   256 $DEF     UCB$L_XM_LSTPRT .BLKL   1        ; Last port value
                     014C   257 $DEF     UCB$L_XM_LSTCSR .BLKL   1        ; Last CSR value
                     0150   258
                     0150   259         $VIELD  UCB,0,<-                 ; XMDRIVER UCB$W_DEVSTS bits
                     0150   260                 <XM_FORK_XMT,,M>,-        ; Transmit fork block in use
                     0150   261                 <,25,-                   ; reserved
                     0150   262                 <XM_INITED,,M>,-          ; Unit initialized
                     0150   263                 <,75,-                   ; reserved
                     0150   264                 <XM_NOTIF,,M>,-           ; Mailbox notified
                     0150   265                 <XM_LOSTERR,,M>,-         ; Unreported fatal error
```

```
                0150   266                 <XM_FORK_PEND,,M>,-     ; Fork process scheduling in progress
                0150   267         >
                0150   268
                0150   269         $VIELD  MOD,0,-                 ; XMDRIVER UCB$L_DEVDEPEND+3 bits
                0150   270         <-                              ;       HARDWARE MODE BITS (byte)
                0150   271                 <XM_HIGH,,M>,-          ; High speed indicator (DMC/DMR)
                0150   272                 <XM_DMC,,M>,-           ; DMC compatible mode (DMR only)
                0150   273                 <XM_INTMOD,,M>,-        ; Integral modem (DMR only)
                0150   274                 <XM_V.35,,M>,-          ; V.35 (DMR only)
                0150   275                 <XM_RS232,,M>,-         ; RS-232C mode (DMR only)
                0150   276                 <XM_RS422,,M>,-         ; RS-422 mode (DMR only)
                0150   277                 <,15>,-                 ; RESERVED
                0150   278                 <XM_BSEL1,,M>,-         ; Indicates that BSEL1 is not locked out
                0150   279         >                               ; ..if set, indicates 1st 2 bits are ok
                0150   280
                0150   281 ;
                0150   282 ; DMC11/DMR11 device register definitions
                0150   283 ;
00000000        0150   284 . = 0
                0000   285 $DEF     XM_I_CSR        .BLKW   1       ; Input CSR (SEL 0)
                0002   286         _VIELD  XM_I,0,<-               ;
                0002   287                 <TYPE,2,M>,-            ; Request type
                0002   288                 <RCV,,M>,-              ; Receive buffer flag
                0002   289                 <,2>,-                  ; reserved
                0002   290                 <RQI,,M>,-              ; Request port
                0002   291                 <IEI,,M>,-              ; Port available interrupt enable
                0002   292                 <RDI,,M>,-              ; Port available
                0002   293                 <STEPUP,,M>,-           ; Step microprocessor
                0002   294                 <ROMI,,M>,-             ; ROM IN
                0002   295                 <ROMO,,M>,-             ; ROM OUT
                0002   296                 <LOOPB,,M>,-            ; Internal loopback
                0002   297                 <,2>,-                  ; Maintenance bits
                0002   298                 <MCLR,,M>,-             ; Master clear device
                0002   299                 <RUN,,M>,-              ; Run
                0002   300         >
                0002   301 $DEF     XM_O_CSR        .BLKW   1       ; Output CSR (SEL 2)
                0004   302         _VIELD  XM_O,0,<-               ;
                0004   303                 <TYPE,2,M>,-            ; Output type
                0004   304                 <RCV,,M>,-              ; Receive buffer flag
                0004   305                 <,3>,-                  ; reserved
                0004   306                 <IEO,,M>,-              ; Output interrupt enable
                0004   307                 <RDO,,M>,-              ; Output ready
                0004   308         >
                0004   309 $DEF     XM_PORT         .BLKW   1       ; Data port register (SEL 4)
                0006   310 $DEF     XM_UCODE        .BLKW   1       ; Data/error port register (SEL 6)
                0008   311         _VIELD  XM_E,0,<-               ;
                0008   312                 <DCHK,,M>,-             ; Data check
                0008   313                 <TIMO,,M>,-             ; Timeout
                0008   314                 <NOBUF,,M>,-            ; Data overrun
                0008   315                 <MOP,,M>,-              ; MOP message received
                0008   316                 <LOST,,M>,-             ; Lost data
                0008   317                 <TRNER,,M>,-            ; Transfer error
                0008   318                 <LINEDWN,,M>,-          ; Line down
                0008   319                 <START,,M>,-            ; Start received
                0008   320                 <NONEXMEM,,M>,-         ; Non-existent memory
                0008   321                 <PROCERR,,M>,-          ; Procedure error
                0008   322                 <POWER,,M>,-            ; System powerfailure (set by driver)
```

```
                     0008   323                 <TIMEOUT,,M>,-          ; Transmit timeout (set by driver)
                     0008   324                 >
                     0008   325 ;
                     0008   326 ; Receive buffer definition
                     0008   327 ;
                     0008   328         $DEFINI RCV
00000000             0000   329 .= 0
                     0000   330 $DEF     RCV_L_LINK      .BLKL   2       ; Forward and backward queue links
                     0008   331 $DEF     RCV_W_BLKSIZE   .BLKW   1       ; Total block size
                     000A   332 $DEF     RCV_B_BLKTYPE   .BLKB   1       ; Block type
                     000B   333 $DEF     RCV_B_MAPSLOT   .BLKB   1       ; Mapping slot number
                     000C   334 $DEF     RCV_L_BACC      .BLKL   1       ; Buffer address / character count
00000048             0010   335         .IIF LT .-CXB$C_HEADER, .=CXB$C_HEADER ; (allow for CXB header)
                     0048   336 $DEF     RCV_T_DATA                      ; Receive data
                     0048   337
                     0048   338         $DEFEND RCV
                     0008   339
                     0008   340 ;
                     0008   341 ; Basetable block definition
                     0008   342 ;
                     0008   343         $DEFINI BAS
00000000             0000   344 .=0
                     0000   345 $DEF     BAS_Q_SPARE     .BLKQ   1       ; Spare quadword
                     0008   346 $DEF     BAS_W_SIZE      .BLKW   1       ; Block size
                     000A   347 $DEF     BAS_B_TYPE      .BLKB   1       ; Block type
                     000B   348 $DEF     BAS_B_SPARE     .BLKB   1       ; Spare byte
                     000C   349 $DEF     BAS_T_DATA                      ; Start of real basetable
                     000C   350 $DEF     BAS_C_HEADER                    ; Size of base table header
                     000C   351
                     000C   352         $DEFEND BAS
```

```
                        .SBTTL  Standard Driver Tables
                ;
                ; Driver Prologue Table
                ;
                DPTAB   -                               ;
                        END=XM_END,-                    ; End of driver
                        ADAPTER=UBA,-                   ; UNIBUS device
                        UCBSIZE=UCB$C_XM_LENGTH,-       ; UCB size
                        NAME=XMDRIVER                   ; Driver name

                DPT_STORE INIT                          ; Initialization data
                DPT_STORE UCB,UCB$B_FIPL,B,8            ; Fork IPL
                DPT_STORE UCB,UCB$B_DIPL,B,21           ; Device IPL
                DPT_STORE UCB,UCB$L_DEVCHAR,L,<-        ; Device characteristics
                        DEV$M_NET!DEV$M_AVL!DEV$M_IDV!DEV$M_ODV>
                DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM ; Device class
                DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_DMC11 ; Assume a DMC11
                DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,256     ; Default buffer size

                DPT_STORE REINIT                        ; Initialization data also for reload
                DPT_STORE DDB,DDB$L_DDT,D,XM$DDT        ; Driver dispatch table
                DPT_STORE CRB,CRB$L_INTD+4,D,PORT_INTR  ; Port interrupt service routine
                DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,UNIT_INIT ; Unit init routine
                DPT_STORE CRB,CRB$L_INTD2+4,D,CONTROL_INTR ; Control interrupt service
                DPT_STORE END

                ;
                ; Driver Dispatch Table
                ;
                DDTAB   DEVNAM=XM,-                     ; Device name
                        START=STARTIO,-                 ; Start I/O routine
                        FUNCTB=FUNCTABLE,-              ; Function decision table
                        CANCEL=CANCEL,-                 ; Cancel I/O routine
                        REGDMP=REGDUMP,-                ; Register dump routine
                        DIAGBF=<32+36>,-                ; Diagnostic buffer size
                        ALTSTART=ALTFDT                 ; Alternate transmit/receive routine
                ;
                ; Function Decision Table
                ;
FUNCTABLE:
                FUNCTAB,-                                               ; Legal functions
                        <WRITEVBLK,WRITELBLK,WRITEPBLK,-                ; Transmit functions
                         READVBLK,READLBLK,READPBLK,-                   ; Receive functions
                         SETMODE,SETCHAR,-                              ; Set mode functions
                         SENSEMODE,SENSECHAR>                           ; Read and/or clear counters
                FUNCTAB,-                                               ; Buffered I/O functions
                        <READLBLK,READPBLK,READVBLK,-                   ;
                         SETMODE,SETCHAR>                               ;
                FUNCTAB XMTFDT,-                                        ; Transmit function dispatcher
                        <WRITELBLK,WRITEPBLK,WRITEVBLK>                 ;
                FUNCTAB RCVFDT,-                                        ; Receive function dispatcher
                        <READLBLK,READPBLK,READVBLK>                    ;
                FUNCTAB SETMODEFDT,-                                    ; Set mode function dispatcher
                        <SETMODE,SETCHAR>                               ;
                FUNCTAB SENSEMODEFDT,-                                  ; Sense mode function dispatcher
                        <SENSEMODE,SENSECHAR>                           ;
```

```
                     0078   411 ;
                     0078   412 ; Counter ID and format table
                     0078   413 ;
                     0078   414 ;       Note: the order of this table is related to the UCB counters.
                     0078   415 ;
           00000000  0078   416 CNT_BUFSIZ = 0
                     0078   417 CNTTAB:                                          ; Counters maintained by driver
                     0078   418         COUNTER BRC,NO, 32                      ;  Bytes received
                     007A   419         COUNTER BSN,NO, 32                      ;  Bytes sent
                     007C   420         COUNTER DBR,NO, 32                      ;  Data blocks received
                     007E   421         COUNTER DBS,NO, 32                      ;  Data blocks sent
                     0080   422                                                 ; Counters maintained by device
                     0080   423         COUNTER DEO,YES,8, 4,HCER,5,DCER        ;  Data errors outbound
                     0087   424         COUNTER DEI,YES,8, 7,HCES,8,DCES        ;  Data errors inbound
                     008E   425         COUNTER LBE,YES,8, 6,NBFS               ;  Local buffer errors
                     0093   426         COUNTER RBE,YES,8, 3,NBFR               ;  Remote buffer errors
                     0098   427         COUNTER LRT,NO, 8, 9,REPS               ;  Local reply timeouts
                     009D   428         COUNTER RRT,NO, 8,10,REPR               ;  Remote reply timeouts
                0000 00A2   429         .WORD   0
                     00A4   430
```

```
                                00A4   432          .SBTTL  UNIT_INIT - Initialize the device unit
                                00A4   433 ;++
                                00A4   434 ; UNIT_INIT - Initialize the device unit
                                00A4   435 ;
                                00A4   436 ; FUNCTIONAL DESCRIPTION:
                                00A4   437 ;
                                00A4   438 ; This routine is called when the driver is loaded and during powerfailure
                                00A4   439 ; recovery.  It sets the unit status to ONLINE.  Also, if called during
                                00A4   440 ; powerfail recovery, it shuts down the device.
                                00A4   441 ;
                                00A4   442 ; INPUTS:
                                00A4   443 ;
                                00A4   444 ;       R3 = CSR address
                                00A4   445 ;       R4 = CSR address
                                00A4   446 ;       R5 = UCB address
                                00A4   447 ;
                                00A4   448 ; OUTPUTS:
                                00A4   449 ;
                                00A4   450 ;       R0,R1,R2,R3,R4,R5 preserved
                                00A4   451 ;--
                                00A4   452 UNIT_INIT:                                   ; Initialize the unit
          64 A5    10    A8    00A4   453          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)  ; Set software status ONLINE
0143 C5     0B A5    90    00A8   454          MOVB    UCB$B_FIPL(R5),UCB$B_XM_FKB+-  ; Set FORK BLOCK FORK IPL
                                00AE   455                  FKB$B_FIPL(R5)          ;
      63   4000 8F    B0    00AE   456          MOVW    #XM_I_M_MCLR,(R3)           ; Master clear the controller
              0E62    30    00B3   457          BSBW    DISABLE_MODEM               ; Disable the modem
   14 64 A5    05    E1    00B6   458          BBC     #UCB$V_POWER,UCB$W_STS(R5),10$ ; Br if not powerfail recovery
               0B    E1    00BB   459          BBC     #XM$V_STS_ACTIVE,-          ; Br if not previously active
        0F 44 A5          00BD   460                  UCB$L_DEVDEPEND(R5),10$     ;
                1F    BB    00C0   461          PUSHR   #^M<R0,R1,R2,R3,R4>         ; Save all registers
53    01        1A    78    00C2   462          ASHL    #XM_E_V_POWER+16,#1,R3      ; Indicate powerfail
54    01        10    78    00C6   463          ASHL    #XM_O_V_TYPE+16,#1,R4       ; Indicate error
              0ABE    30    00CA   464          BSBW    SCHED_FORK                  ; Schedule fork process
                1F    BA    00CD   465          POPR    #^M<R0,R1,R2,R3,R4>         ; Restore registers
                      05    00CF   466 10$:     RSB                                 ;
                            00D0   467
```

```
                                00D0   469          .SBTTL XMTFDT - Transmit I/O FDT routine
                                00D0   470 ;++
                                00D0   471 ; XMTFDT - Transmit I/O FDT routine
                                00D0   472 ;
                                00D0   473 ; Functional description:
                                00D0   474 ;
                                00D0   475 ; This routine is called by the SYS$QIO service to dispatch a WRITE I/O
                                00D0   476 ; request.
                                00D0   477 ;
                                00D0   478 ; The QIO parameters for WRITES are:
                                00D0   479 ;
                                00D0   480 ;       P1 = address of the buffer
                                00D0   481 ;       P2 = size of the buffer
                                00D0   482 ;       P3-P6 = (unused)
                                00D0   483 ;
                                00D0   484 ; The buffer is validated for access and locked into the caller's working
                                00D0   485 ; set, a transmit UNIBUS map register set is allocated, the buffer
                                00D0   486 ; is mapped, the device input port is requested, the buffer address and size
                                00D0   487 ; are passed to the device, and finally the I/O request is queued to await
                                00D0   488 ; the completion of the transmit by the device.
                                00D0   489 ;
                                00D0   490 ; If no transmit slot or mapping registers are available, put the I/O request
                                00D0   491 ; into a wait queue.  When a transmit in progress completes, it will restart
                                00D0   492 ; the waiting request.  Note - this design depends on having at least one set
                                00D0   493 ; of map registers pre-allocated.
                                00D0   494 ;
                                00D0   495 ; For requests specifing IO$M_ENABLMBX the attention mailbox is enabled.
                                00D0   496 ;
                                00D0   497 ; Inputs:
                                00D0   498 ;
                                00D0   499 ;       R0-R2 = scratch registers
                                00D0   500 ;       R3 = I/O packet address
                                00D0   501 ;       R4 = PCB address
                                00D0   502 ;       R5 = UCB address
                                00D0   503 ;       R6 = CCB address
                                00D0   504 ;       R7 = bit number of the I/O function code
                                00D0   505 ;       R8 = address of the FDT table entry for this routine
                                00D0   506 ;       R9-R11 = scratch registers
                                00D0   507 ;       AP = address of first QIO parameter
                                00D0   508 ;
                                00D0   509 ; Outputs:
                                00D0   510 ;
                                00D0   511 ;       R0 = status of transmit request initiation
                                00D0   512 ;
                                00D0   513 ;       R3,R5 are preserved.
                                00D0   514 ;--
                                00D0   515 XMTFDT:                                      ; Transmit FDT routine
     50    14    3C             00D0   516          MOVZWL  S^#SS$_BADPARAM,R0         ; Assume bad buffer parameters
   51    04 AC    3C             00D3   517          MOVZWL  P2(AP),R1                  ; Get buffer size
             2F    13             00D7   518          BEQL    ABORTIO                    ; Br if zero - abort I/O
3FFF 8F    51    B1             00D9   519          CMPW    R1,#MAX_C_BUFSIZE          ; Is buffer too big?
             28    1A             00DE   520          BGTRU   ABORTIO                    ; Br if yes - abort I/O
     50    6C    D0             00E0   521          MOVL    P1(AP),R0                  ; Get user buffer virtual address
 00000000'GF    16             00E3   522          JSB     G^EXE$WRITELOCK            ; Check buffer access and lock down
                               00E9   523                                             ; (no return means no access)
                               00E9   524          SETIPL  UCB$B_FIPL(R5)             ; Synch access to UCB
50   0084 8F    3C             00ED   525          MOVZWL  #SS$_DEVOFFLINE,R0         ; Assume device is not active
```

```
                 0B   E1  00F2   526          BBC     #XM$V_STS_ACTIVE,-      ; Br if not active - abort I/O
           11 44 A5       00F4   527                  UCB$L_DEVDEPEND(R5),ABORTIO
                 07   E1  00F7   528          BBC     #IO$V_ENABLMBX,-        ; Br if mailbox not to be enabled
           04 20 A3       00F9   529                  IRP$W_FUNC(R3),5$       ;
         44 A5    10   A8  00FC   530          BISW    #XM$M_CHR_MBX,UCB$L_DEVDEPEND(R5) ; Enable mailbox
                 0C   10  0100   531 5$:       BSBB    XMT_START               ; Start transmit operation
        00000000'GF   17  0102   532          JMP     G^EXE$QIORETURN         ; Exit QIO service to await completion
                       0108   533
                       0108   534 ABORTIO:                                   ; Abort the I/O request
        00000000'GF   17  0108   535          JMP     G^EXE$ABORTIO           ; and exit QIO service
                       010E   536
                       010E   537 ;
                       010E   538 ; Start transmit operation.
                       010E   539 ;
                       010E   540 XMT_START:                                 ; Start transmit operation
      0094 D5    63   0E  010E   541          INSQUE  (R3),@UCB$Q_XM_XMT_REQ+4(R5) ; Insert at end of wait queue
                       0113   542
                       0113   543 XMT_START_ALT:                             ; Alternate entry to start xmits
                 00   E0  0113   544          BBS     #UCB$V_XM_FORK_XMT,-    ; Br if XMT fork block in use
           1A 68 A5       0115   545                  UCB$W_DEVSTS(R5),10$    ;
      53   0090 D5   0F  0118   546          REMQUE  @UCB$Q_XM_XMT_REQ(R5),R3 ; Remove first entry from queue
                 13   1D  011D   547          BVS     10$                     ; Br if none
                       011F   548 ;
                       011F   549 ; Find a free mapping register slot.  If none currently available, put
                       011F   550 ; the I/O request in the wait queue.
                       011F   551 ;
      54   010B C5   9A  011F   552          MOVZBL  UCB$B_XM_XMT_MAX(R5),R4 ; Get max concurrent transmits
54    0109 C5   54   00   EB  0124   553          FFC     #0,R4,UCB$B_XM_XMT_MAP(R5),R4 ; Find a free transmit slot
                 06   12  012B   554          BNEQ    20$                     ; Br if one free
      0090 C5    63   0E  012D   555          INSQUE  (R3),UCB$Q_XM_XMT_REQ(R5) ; Re-insert request in wait queue
                 05   0132   556 10$:       RSB                             ; Return to caller
                       0133   557
                       0133   558 ;
                       0133   559 ; Allocate UNIBUS map registers
                       0133   560 ;
                       0133   561 20$:       ASSUME  IRP$W_BOFF+2 EQ IRP$W_BCNT
                       0133   562           ASSUME  UCB$W_BOFF+2 EQ UCB$W_BCNT
7C A5    30 A3   D0   0133   563          MOVL    IRP$W_BOFF(R3),UCB$W_BOFF(R5) ; Set buffer offset and count
78 A5    2C A3   D0   0138   564          MOVL    IRP$L_SVAPTE(R3),UCB$L_SVAPTE(R5) ; Set buffer PTE address
            01   A8  013D   565          BISW    #UCB$M_XM_FORK_XMT,-    ; Assume we will have to wait
         68 A5       013F   566                  UCB$W_DEVSTS(R5)        ; ..for fork block
  0000014F'EF   9F  0141   567          PUSHAB  30$                     ; Push address of fork process
  00000000'GF   17  0147   568          JMP     G^IOC$REQMAPREG         ; Request map registers
                       014D   569 ;
                       014D   570 ; The following code may be executed as a fork process, therefore
                       014D   571 ; we must provide for a timeout service routine address.
                       014D   572 ;
              0BB1'    014D   573          .WORD   TIMEOUT-.               ; Offset to timeout routine
            01   AA  014F   574 30$:       BICW    #UCB$M_XM_FORK_XMT,-    ; Fork block is no longer in use
         68 A5       0151   575                  UCB$W_DEVSTS(R5)        ;
            0B   E0  0153   576          BBS     #XM$V_STS_ACTIVE,-      ; Br if still active
      0C 44 A5       0155   577                  UCB$L_DEVDEPEND(R5),40$ ;
                       0158   578          RELMPR                          ; Else, release the map registers
       50   2C   3C   015E   579          MOVZWL  #SS$_ABORT,R0           ; Return request in error
          0B0C   31   0161   580          BRW     IO_DONE                 ; Complete the I/O request
                       0164   581
            57   DD   0164   582 40$:       PUSHL   R7                      ; Save R7
```

```
                57   24 A5    D0  0166   583          MOVL    UCB$L_CRB(R5),R7        ; Get CRB address
                                 016A   584          ASSUME  VEC$W_MAPREG+2 EQ VEC$B_NUMREG
                                 016A   585          ASSUME  VEC$B_NUMREG+1 EQ VEC$B_DATAPATH
                     34 A7    D0  016A   586          MOVL    CRB$L_INTD+VEC$W_MAPREG(R7),- ; Save mapping info
                 00EC C544        016D   587                  UCB$L_XM_XMT_MAP(R5)[R4]
                                  0171   588 ;
                                  0171   589 ; Map the buffer
                                  0171   590 ;
                                  0171   591          SETBIT  R4,UCB$B_XM_XMT_MAP(R5)         ; Set mapping slot in use flag
                3C A3    54      90  0177   592          MOVB    R4,IRP$L_MEDIA+4(R3)            ; Save mapping slot number used
                                 017B   593          ASSUME  IRP$W_BOFF+2 EQ IRP$W_BCNT
             38 A3    30 A3    D0  017B   594          MOVL    IRP$W_BOFF(R3),IRP$L_MEDIA(R3)  ; Move byte offset and size
                     34 A7    F0  0180   595          INSV    CRB$L_INTD+VEC$W_MAPREG(R7),-   ; Insert BA9-BA15
          38 A3    07    09      0183   596                  #9,#7,IRP$L_MEDIA(R3)           ;
    50    34 A7    02    07  EF  0187   597          EXTZV   #7,#2,CRB$L_INTD+VEC$W_MAPREG(R7),R0 ; Get BA16-BA17
38 A3     02    1E    50      F0  018D   598          INSV    R0,#30,#2,IRP$L_MEDIA(R3)       ; Insert BA16-BA17
          00000000'GF    16      0193   599          JSB     G^IOC$LOADUBAMAPA               ; Load map registers
                                  0199   600 ;
                                  0199   601 ; Request and load the port with the buffer address and size, and return.
                                  0199   602 ;
                      57 8ED0     0199   603          POPL    R7                      ; Restore R7
                                  019C   604          DSBINT  UCB$B_DIPL(R5)          ; Synch access to device
                    07E1    30    01A3   605          BSBW    LOAD_PORT               ; Load port
                                  01A6   606          ENBINT                          ; Restore IPL
                            05    01A9   607          RSB                             ; Return to caller to await completion
                                  01AA   608
```

```
                                                H 2
                                .SBTTL  RCVFDT - Receive I/O FDT routine
                        ;++
                        ; RCVFDT - Receive I/O FDT routine
                        ;
                        ; Functional description:
                        ;
                        ; This routine is called by the SYS$QIO service to dispatch a READ I/O
                        ; request.
                        ;
                        ; The QIO parameters for READs are:
                        ;
                        ;       P1 = address of the buffer
                        ;       P2 = size of the buffer
                        ;       P3-P6 = (unused)
                        ;
                        ; The specified buffer is checked for accessibility. The buffer address and
                        ; count are saved in the packet. Then IPL is set to device fork IPL and if
                        ; a message is available the operation is completed.  Otherwise the packet
                        ; is queued onto the waiting receive list. The mailbox notified bit is cleared.
                        ;
                        ; For requests specifing IO$M_NOW, the I/O is completed with status of
                        ; SS$_ENDOFILE if no message is available when the test is made.
                        ;
                        ; For requests specifing IO$M_DSABLMBX the attention mailbox is disabled.
                        ;
                        ; Inputs:
                        ;
                        ;       R3 = I/O packet address
                        ;       R4 = PCB address
                        ;       R5 = UCB address
                        ;       R6 = CCB address
                        ;       R7 = Function code
                        ;       AP = Address of first I/O request parameter
                        ;
                        ; Outputs:
                        ;
                        ;       R0 = Status of the receive request
                        ;
                        ;       R3-R7 preserved.
                        ;--
                        RCVFDT:                                         ; Receive function routine
    50    14    3C      MOVZWL  S^#SS$_BADPARAM,R0                    ; Assume illegal size
51    04    AC    3C      MOVZWL  P2(AP),R1                             ; Get size
            2F    13      BEQL    10$                                   ; Br if none specified
    50    6C    D0      MOVL    P1(AP),R0                             ; Get buffer address
38  A3      50    D0      MOVL    R0,IRP$L_MEDIA(R3)                    ; Save address
      30    A3    B4      CLRW    IRP$W_BOFF(R3)                        ; No quota to return during completion
00000000'GF       16      JSB     G^EXE$READCHK                         ; Check buffer accessibility
                  01C3    58                                            ; (no return on no access)
                  01C3    59      SETIPL  UCB$B_FIPL(R5)                ; Synchronize access to the UCB
50    0084  8F    3C      01C7    60      MOVZWL  #SS$_DEVOFFLINE,R0    ; Assume device not active
            0B    E1      01CC    61      BBC     #XM$V_STS_ACTIVE,-    ; Br if not active - abort I/O
                  11 44 A5        01CE    62              UCB$L_DEVDEPEND(R5),10$
            0A    E1      01D1    63      BBC     #IO$V_DSABLMBX,-      ; Br if not disabling mailbox
                  04 20 A3        01D3    64              IRP$W_FUNC(R3),5$
                  44 A5   10  AA  01D6    65      BICW    #XM$M_CHR_MBX,UCB$L_DEVDEPEND(R5) ; Else, disable mailbox
                  09      10      01DA    66 5$:  BSBB    RCV_START             ; Start receive operation
```

```
                00000000'GF   17  01DC   667          JMP     G^EXE$QIORETURN          ; Return to await completion
                              01E2   668
                      FF23    31  01E2   669 10$:      BRW     ABORTIO                  ; Abort the I/O request
                              01E5   670 ;
                              01E5   671 ; Start receive operation.
                              01E5   672 ;
                              01E5   673 RCV_START:                                   ; Start receive operation
        68 A5         0800 8F   AA  01E5   674          BICW    #UCB$M_XM_NOTIF,UCB$W_DEVSTS(R5) ; Clear notified status
                              01EB   675 ;
                              01EB   676 ; Check for message available and complete receive if it is
                              01EB   677 ;
              52      00C8 D5   0F  01EB   678          REMQUE  @UCB$Q_XM_RCV_MSG(R5),R2 ; Dequeue a received message
                            03  1D  01F0   679          BVS     15$                      ; Br if none
                          0A43  31  01F2   680          BRW     FINISH_RCV_IO            ; Complete the I/O and exit
                              01F5   681 ;
                              01F5   682 ; Queue the requst for future message arrival unless IO$M_NOW specified.
                              01F5   683 ;
        06 20 A3        06    E0  01F5   684 15$:      BBS     #IO$V_NOW,IRP$W_FUNC(R3),20$ ; Br if read NOW
         009C D5        63    0E  01FA   685          INSQUE  (R3),@UCB$Q_XM_RCV_REQ+4(R5) ; Queue the I/O packet
                              05  01FF   686          RSB                              ;
                              0200   687
              50      0870 8F   3C  0200   688 20$:      MOVZWL  #SS$_ENDOFFILE,R0        ; Set no message status
                          0A68  31  0205   689          BRW     IO_DONE                  ; Complete the I/O and exit
                              0208   690
                              0208   691
```

```
                                                J 2
                     0208   693          .SBTTL  ALTFDT - Alternate Transmit/Receive I/O routine
                     0208   694 ;++
                     0208   695 ; ALTFDT - Alternate Transmit/Receive dispatch routine
                     0208   696 ;
                     0208   697 ; Functional description:
                     0208   698 ;
                     0208   699 ; This routine is called by the other drivers to pass an "internal" I/O
                     0208   700 ; request to the driver.  "Internal" IRP's are not built via $QIO.
                     0208   701 ; The action here is to setup the IRP fields as if the packet had been
                     0208   702 ; processed by the FDT routines.
                     0208   703 ;
                     0208   704 ; In this driver, the alternate entry point is called by the DECnet
                     0208   705 ; Transport layer driver.
                     0208   706 ;
                     0208   707 ; Inputs:
                     0208   708 ;
                     0208   709 ;       R3 = I/O packet address
                     0208   710 ;       R5 = UCB address
                     0208   711 ;
                     0208   712 ;       All pertinent fields of the IRP are assumed to be valid.
                     0208   713 ;
                     0208   714 ;       IPL = FIPL
                     0208   715 ;
                     0208   716 ; Outputs:
                     0208   717 ;
                     0208   718 ;       R0 = Success
                     0208   719 ;
                     0208   720 ;       R3 and R5 preserved.
                     0208   721 ;--
                     0208   722 ALTFDT:                                      ; Alternate FDT routine
50   0084 8F   3C    0208   723          MOVZWL  #SS$_DEVOFFLINE,R0          ; Assume device not active
          0B   E0    020D   724          BBS     #XM$V_STS_ACTIVE,-          ; Br if active
03 44 A5              020F   725                  UCB$L_DEVDEPEND(R5),5$      ;
       0A5B   31      0212   726          BRW     IO_DONE                     ; Post the I/O request in error
                     0215   727
          01   E0    0215   728 5$:       BBS     #IRP$V_FUNC,-               ; Br if receive function
05 2A A3              0217   729                  IRP$W_STS(R3),10$           ;
       FEF1   30      021A   730          BSBW    XMT_START                   ; Initiate the transmit
          0E   11      021D   731          BRB     20$                         ;
                     021F   732
52   2C A3   D0      021F   733 10$:       MOVL    IRP$L_SVAPTE(R3),R2         ; Get address of input buffer
          06   13      0223   734          BEQL    15$                         ; Br if none
       06A6   30      0225   735          BSBW    ADDRCVLIST                  ; Add it to the receive list
     2C A3   D4      0228   736          CLRL    IRP$L_SVAPTE(R3)            ; Buffer now used
          B8   10      022B   737 15$:       BSBB    RCV_START                   ; Initiate the receive
                     022D   738
50   01   3C      022D   739 20$:       MOVZWL  S^#SS$_NORMAL,R0            ; Always return success
          05   0230   740          RSB                                 ;
                     0231   741
                     0231   742
```

```
0231    744          .SBTTL  SETMODEFDT - Set mode I/O operation FDT dispatch routine
0231    745 ;++
0231    746 ; SETMODEFDT - Set mode FDT processing
0231    747 ;
0231    748 ; Functional description:
0231    749 ;
0231    750 ; This routine is called by the SYS$QIO service to dispatch a SETMODE/SETCHAR
0231    751 ; I/O request.
0231    752 ;
0231    753 ; The QIO parameters for SETMODE or SETCHAR are:
0231    754 ;
0231    755 ;       P1 = address of 8 byte characteristics buffer
0231    756 ;       P2 = (unused)
0231    757 ;       P3 = number of receive buffers to pre-allocate (IO$M_STARTUP only)
0231    758 ;       P4-P6 = (unused)
0231    759 ;
0231    760 ;  No modifier -
0231    761 ;
0231    762 ;       This function is done in the STARTIO routine. Control is passed to
0231    763 ;       EXESSETMODE to validate the new mode buffer and queue the packet.
0231    764 ;
0231    765 ;  IO$M_CTRL -
0231    766 ;
0231    767 ;       Perform this function on the LINE rather than the circuit. The only
0231    768 ;       extra action that is done, is that on a STARTUP request the modem
0231    769 ;       is enabled via a master clear to the DMC. This will re-enable the
0231    770 ;       DTR signal to the modem. On a SHUTDOWN request, the DTR signal is
0231    771 ;       inhibited. The STARTUP or SHUTDOWN bit is then cleared and the I/O
0231    772 ;       request is processed as a regular request for the CIRCUIT.
0231    773 ;
0231    774 ;  IO$M_STARTUP -
0231    775 ;
0231    776 ;       This function starts the unit and sets the mode.
0231    777 ;       The action here is to pick up the user buffered i/o quota
0231    778 ;       and allocate the base table. The base table address is saved in
0231    779 ;       IRP$L_SVAPTE. The quota is taken from the user is in IRP$W_BOFF.
0231    780 ;       This value will be the IOSB+2 value at I/O done. This function is
0231    781 ;       complete when the base table has been given to the unit. The mailbox
0231    782 ;       is enabled and a receive is started.  This function is done partially
0231    783 ;       here and the remainder is done in STARTIO.
0231    784 ;
0231    785 ;  IO$M_SHUTDOWN -
0231    786 ;
0231    787 ;       This function shuts down the unit and optionally resets the mode.
0231    788 ;       A cancel I/O is preformed, all outstanding I/O is completed, the base
0231    789 ;       table and message blocks are all returned and the unit is left in an
0231    790 ;       idle state. This function cannot be done here and the FDT processing is
0231    791 ;       that of all setmode operations.
0231    792 ;
0231    793 ;  IO$M_ATTNAST -
0231    794 ;
0231    795 ;       This function sets up a AST to be delivered on one of the following
0231    796 ;       conditions:
0231    797 ;
0231    798 ;               Fatal error that caused shutdown.
0231    799 ;               Message available to be received.
0231    800 ;
```

```
                     0231   801 ;
                     0231   802 ; Inputs:
                     0231   803 ;
                     0231   804 ;       R3 = I/O packet address
                     0231   805 ;       R4 = PCB address
                     0231   806 ;       R5 = UCB address
                     0231   807 ;       R6 = CCB address
                     0231   808 ;       R7 = function code
                     0231   809 ;       AP = Address of first I/O request parameter
                     0231   810 ;
                     0231   811 ; Outputs:
                     0231   812 ;
                     0231   813 ;       R0 = Status of setmode request
                     0231   814 ;
                     0231   815 ;       R3-R5 preserved.
                     0231   816 ;--
                     0231   817 SETMODEFDT:                                ; Set mode FDT processing
        2C A3    D4  0231   818         CLRL    IRP$L_SVAPTE(R3)          ; Set no buffer
57      20 A3    B0  0234   819         MOVW    IRP$W_FUNC(R3),R7         ; Get entire function code
06 57    09    E1  0238   820         BBC     #IO$V_CTRL,R7,5$          ; Br if not a LINE request
                     023C   821 ;
                     023C   822 ; LINE request
                     023C   823 ;
         00E2    30  023C   824         BSBW    SETMODEFDT_LINE           ; Process a LINE request
        3A 50    E8  023F   825         BLBS    R0,10$                    ; Br if request is complete
                     0242   826
40 57    08    E1  0242   827 5$:     BBC     #IO$V_ATTNAST,R7,20$      ; Br if not AST request
                     0246   828 ;
                     0246   829 ; Attention AST request
                     0246   830 ;
57    0114 C5    9E  0246   831         MOVAB   UCB$L_XM_AST(R5),R7       ; Set address of AST block listhead
 00000000'GF    16  024B   832         JSB     G^COM$SETATTNAST          ; Create AST block
                     0251   833         DSBINT  UCB$B_FIPL(R5)            ; Synch access to UCB
           54    D4  0258   834         CLRL    R4                        ; Set Mailbox msg
           0C    E5  025A   835         BBCC    #UCB$V_XM_LOSTERR,-       ; Br unless unreported fatal errors
     06 68 A5         025C   836                 UCB$W_DEVSTS(R5),7$       ;
   54    00'8F    9A  025F   837         MOVZBL  #MSG$_XM_SHUTDN,R4        ; Set message code
           0F    11  0263   838         BRB     8$
           03    E1  0265   839 7$:     BBC     #UCB$V_XM_INITED,-        ; Br if device not initialized
     12 68 A5         0267   840                 UCB$W_DEVSTS(R5),10$      ;
51    00C8 C5    9E  026A   841         MOVAB   UCB$Q_XM_RCV_MSG(R5),R1   ; Get address received message queue
     61    51    D1  026F   842         CMPL    R1,(R1)                   ; Any messages in queue?
           08    13  0272   843         BEQL    10$                       ; Br if no - nothing to report yet
           53    DD  0274   844 8$:     PUSHL   R3                        ; Save I/O packet address
         0A41    30  0276   845         BSBW    POKE_USER                 ; Deliver the AST immediately
           53  8ED0  0279   846         POPL    R3                        ; Restore register
 51    44 A5    D0  027C   847 10$:    MOVL    UCB$L_DEVDEPEND(R5),R1    ; Get device characteristics
 00000000'GF    17  0280   848         JMP     G^EXE$FINISHIO            ; Complete the I/O
                     0286   849 ;
                     0286   850 ; Set mode, startup, or shutdown request.  Get the characteristics buffer.
                     0286   851 ;
        38 A3    7C  0286   852 20$:    CLRQ    IRP$L_MEDIA(R3)           ; Reset mode data buffer
           53    DD  0289   853         PUSHL   R3                        ; Save I/O packet address
     52    6C    D0  028B   854         MOVL    P1(AP),R2                 ; Get address of new characteristics
           11    13  028E   855         BEQL    30$                       ; Br if none specified
     50    0C    3C  0290   856         MOVZWL  S^#SS$_ACCVIO,R0          ; Assume no access
                     0293   857         IFNORD  #8,(R2),45$               ; Br if no access to buffer
```

```
                38 A3    62    7D  0299    858          MOVQ    (R2),IRP$L_MEDIA(R3)    ; Save new characteristics in packet
                38 A3    01    90  029D    859          MOVB    #1,IRP$L_MEDIA(R3)      ; Mark it "valid"
                07 57    06    E0  02A1    860 30$:     BBS     #IO$V_STARTUP,R7,50$    ; Br if startup function
                         53  8ED0  02A5    861          POPL    R3                      ; Restore packet address
                         6B    11  02A8    862          BRB     90$                     ; Queue the packet
                         6F    11  02AA    863 45$:     BRB     100$                    ;
                                   02AC    864 ;
                                   02AC    865 ; Startup request - check caller's quota and allocate the basetable.
                                   02AC    866 ;
                51    08 AC    9A  02AC    867 50$:     MOVZBL  P3(AP),R1               ; Get number of receives to preallocate
                         52    D5  02B0    868          TSTL    R2                      ; Any characteristics specified?
                         04    12  02B2    869          BNEQ    55$                     ; Br if yes
                52    40 A5    9E  02B4    870          MOVAB   UCB$B_DEVCLASS(R5),R2   ; Else, set addr to current ones
                52    02 A2    3C  02B8    871 55$:     MOVZWL  2(R2),R2                ; Get receive buffer size
                   50    14    3C  02BC    872          MOVZWL  S^#SS$_BADPARAM,R0      ; Assume bad parameters
                   51    52    C4  02BF    873          MULL    R2,R1                   ; Compute total needed for buffers
                         57    13  02C2    874          BEQL    100$                    ; Br if somehow in error
              51    0100 8F    A0  02C4    875          ADDW    #BASETAB_SIZE,R1        ; Add size of base table
                   57    51    3C  02C9    876          MOVZWL  R1,R7                   ; Copy quota
                   57    51    D1  02CC    877          CMPL    R1,R7                   ; Overflow?
                         4A    12  02CF    878          BNEQ    100$                    ; Br if in error
               00000000'GF    16  02D1    879          JSB     G^EXE$BUFQUOPRC         ; Check caller's quota
                      41 50    E9  02D7    880          BLBC    R0,100$                 ; Br if error
              51    010C 8F    3C  02DA    881          MOVZWL  #BASETAB_SIZE+BAS_C_HEADER,R1 ; Set size of basetable + header
               00000000'GF    16  02DF    882          JSB     G^EXE$ALLOCBUF          ; Allocate the table
                      33 50    E9  02E5    883          BLBC    R0,100$                 ; Return if error
                         53  8ED0  02E8    884          POPL    R3                      ; Restore I/O packet address
                30 A3    57    B0  02EB    885          MOVW    R7,IRP$W_BOFF(R3)       ; Save quota in packet
                   57    57    3C  02EF    886          MOVZWL  R7,R7                   ; Convert to longword
                50    0080 C4    D0  02F2    887          MOVL    PCB$L_JIB(R4),R0        ; Get job info block address
                   20 A0    57    C2  02F7    888          SUBL    R7,JIB$L_BYTCNT(R0)     ; Adjust byte count quota
                   24 A0    57    C2  02FB    889          SUBL    R7,JIB$L_BYTLM(R0)      ; ..and byte limit quota
                   2C A3    52    D0  02FF    890          MOVL    R2,IRP$L_SVAPTE(R3)     ; Save base table data address
                   08 A2    51    D0  0303    891          MOVL    R1,BAS_W_SIZE(R2)       ; Save size of base table
                         38    BB  0307    892          PUSHR   #^M<R3,R4,R5>           ; Save registers
              00    0C A2    00    2C  0309    893          MOVC5   #0,BAS_T_DATA(R2),#0,-  ; Zero the base table
              0C A2    00F4 8F        030E    894                  #BASETAB_SIZE-BAS_T_DATA,BAS_T_DATA(R2) ;
                         38    BA  0313    895          POPR    #^M<R3,R4,R5>           ; Restore registers
                00000000'GF    17  0315    896 90$:     JMP     G^EXE$QIODRVPKT         ; Queue the I/O packet
                                   031B    897 ;
                                   031B    898 ; Setmode/start error
                                   031B    899 ;
                         53  8ED0  031B    900 100$:    POPL    R3                      ; Restore I/O packet address
                      FDE7    31  031E    901          BRW     ABORTIO                 ; Abort the I/O request
                                   0321    902
```

```
                     0321    904          .SBTTL  SETMODEFDT_LINE - Set mode I/O operation FDT routine for LINE
                     0321    905 ;++
                     0321    906 ; SETMODEFDT_LINE - Set mode FDT processing for DMC LINE
                     0321    907 ;
                     0321    908 ; Functional description:
                     0321    909 ;
                     0321    910 ; This routine is called when normal SETMODE FDT processing has detected that
                     0321    911 ; the I/O request is on the line.
                     0321    912 ;
                     0321    913 ; QIO parameters are the same as for regular SETMODE.
                     0321    914 ;
                     0321    915 ; Modifiers:
                     0321    916 ;
                     0321    917 ;   IO$M_STARTUP -
                     0321    918 ;
                     0321    919 ;       This function forces the DMC/DMR to be master cleared to re-enable
                     0321    920 ;       the DTR modem signal.
                     0321    921 ;
                     0321    922 ;   IO$M_SHUTDOWN -
                     0321    923 ;
                     0321    924 ;       This function shuts down the unit's modem, by calling a routine to
                     0321    925 ;       disable the DTR signal to the modem.
                     0321    926 ;
                     0321    927 ; Inputs:
                     0321    928 ;
                     0321    929 ;       R3 = I/O packet address
                     0321    930 ;       R4 = PCB address
                     0321    931 ;       R5 = UCB address
                     0321    932 ;       R6 = CCB address
                     0321    933 ;       R7 = Function code
                     0321    934 ;       AP = Address of first I/O request parameter
                     0321    935 ;
                     0321    936 ;       IPL = IPL$_ASTDEL
                     0321    937 ;
                     0321    938 ; Outputs:
                     0321    939 ;
                     0321    940 ;       R0 = LBC, if we can continue, else all done with request
                     0321    941 ;       R1 is destroyed, all other registers are preserved
                     0321    942 ;
                     0321    943 ;--
                     0321    944 SETMODEFDT_LINE:                            ; Set mode FDT processing for DMC LINE
 50    01    9A      0321    945          MOVZBL  #1,R0                      ; Assume we can't continue
       03    E0      0324    946          BBS     #UCB$V_XM_INITED,-         ; Br if device initialized
    13 68    A5      0326    947                  UCB$W_DEVSTS(R5),10$       ;  ignore request, circuit must be
                     0329    948                                             ;  off before playing with modem.
 16 57    06    E5   0329    949          BBCC    #IO$V_STARTUP,R7,20$       ; Br if not startup function
                     032D    950 ;
                     032D    951 ;       STARTUP LINE request, enable DTR
                     032D    952 ;
 51    24    A5      032D    953          MOVL    UCB$L_CRB(R5),R1           ; Get CRB address
                     0331    954          ASSUME  IDB$L_CSR EQ 0
 51    2C    B1      0331    955          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R1 ; Get CSR address
 61  4000    8F      0335    956          MOVW    #XM_I_M_MCLR,(R1)          ; Master clear controller, resets DTR
       50    D4      033A    957          CLRL    R0                         ; Allow this function to continue
20 A3  02C0    8F    033C    958 10$:     BICW    #IO$M_STARTUP!IO$M_SHUTDOWN!- ; Clear out all processed flags
                     0342    959                  IO$M_CTRL,IRP$W_FUNC(R3) ;
                     0342    960          RSB
```

```
                                0343   961
                F5 57   07   E1 0343   962 20$:    BBC     #IO$V_SHUTDOWN,R7,10$   ; Br if not shutdown function, stop
                                0347   963 ;
                                0347   964 ;       Disable the modem line.
                                0347   965 ;
                          0BCE  30 0347   966       BSBW    DISABLE_MODEM           ; Disable the modem DTR line
                            55  DD 034A   967       PUSHL   R5                      ; Save UCB address
                                034C   968       ASSUME  IRP$L_ARB+4+TQE$C_LENGTH LE IRP$C_LENGTH
        55   53  00000094 8F   C1 034C   969       ADDL3   #IRP$C_LENGTH-TQE$C_LENGTH,R3,R5 ; Use end of IRP as TQE
                  0A A5    0F  90 0354   970       MOVB    #DYN$C_TQE,TQE$B_TYPE(R5) ; Set structure type
                0C A5   96'AF  9E 0358   971       MOVAB   B^30$,TQE$L_FPC(R5)     ; Set wakeup routine address
                  0B A5    01  90 035D   972       MOVB    #TQE$C_SSSNGL,TQE$B_RQTYPE(R5) ; Set the TQE request type
                  10 A5    53  D0 0361   973       MOVL    R3,TQE$L_FR3(R5)        ; Save IRP address in TQE
                                0365   974       DSBINT  #IPL$_TIMER             ; Raise IPL
    50  00000000 000F4240 8F   7D 036B   975       MOVQ    #SHUT_TIME,R0           ; Calculate the delta time
          50 00000000'GF   C0 0376   976       ADDL    G^EXE$GQ_SYSTIME,R0     ;       ...
          51 00000004'GF   D8 037D   977       ADWC    G^EXE$GQ_SYSTIME+4,R1   ;       ...
             00000000'GF   16 0384   978       JSB     G^EXE$INSTIMQ           ; Insert TQE on timer queue
                                038A   979       ENBINT                          ; Restore IPL
                       55 8ED0  038D   980       POPL    R5                      ; Restore UCB address
             00000000'GF   17 0390   981       JMP     G^EXE$QIORETURN         ; Wait for the TQE to complete request
                                0396   982 ;
                                0396   983 ; TQE wakeup routine
                                0396   984 ;
                                0396   985 ;       R3 = IRP address
                                0396   986 ;       R5 = TQE address at end of IRP
                                0396   987 ;
                                0396   988 ;       IPL = IPL$_TIMER
                                0396   989 ;
          50   01   9A 0396   990 30$:    MOVZBL  #SS$_NORMAL,R0          ; Return success
            55   DD   0399   991       PUSHL   R5                      ; Save TQE address
    55    1C A3   D0 039B   992       MOVL    IRP$L_UCB(R3),R5        ; Copy UCB address to R5
          08CE   30 039F   993       BSBW    IO_DONE                 ; Complete the I/O request
            55 8ED0  03A2   994       POPL    R5                      ; Restore TQE address
                05  03A5   995       RSB                             ; Return to caller
                    03A6   996
```

```
                                03A6   998          .SBTTL  SENSEMODE - Sense mode I/O operation FDT
                                03A6   999 ;++
                                03A6  1000 ; SENSEMODE - Sense mode FDT processing
                                03A6  1001 ;
                                03A6  1002 ; This routine is called by the SYS$QIO service to dispatch a SENSEMODE
                                03A6  1003 ; SENSECHAR I/O request.
                                03A6  1004 ;
                                03A6  1005 ; The QIO parameters for SENSEMODE are:
                                03A6  1006 ;
                                03A6  1007 ;       P1 = (unused)
                                03A6  1008 ;       P2 = address of descriptor of buffer to receive counters
                                03A6  1009 ;       P3-P6 = (unused)
                                03A6  1010 ;
                                03A6  1011 ; The error counters are returned to the caller in NICE format in the buffer.
                                03A6  1012 ;
                                03A6  1013 ; Inputs:
                                03A6  1014 ;
                                03A6  1015 ;       R3 = I/O packet address
                                03A6  1016 ;       R4 = PCB address
                                03A6  1017 ;       R5 = UCB address
                                03A6  1018 ;       R6 = CCB address
                                03A6  1019 ;       R7 = Function code
                                03A6  1020 ;       AP = Address of first I/O request parameter
                                03A6  1021 ;
                                03A6  1022 ; Outputs:
                                03A6  1023 ;
                                03A6  1024 ;       R0 = Status of diagnose request
                                03A6  1025 ;
                                03A6  1026 ;       R3-R5 preserved.
                                03A6  1027 ;--
                                03A6  1028 SENSEMODEFDT:                                ; Sense mode FDT routine
                      7D 20 A3  03A6  1029          BBC     #IO$V_RD_COUNT,IRP$W_FUNC(R3),80$ ; Br if not returning counters
                                03AB  1030 ;
                                03AB  1031 ; Check the caller's buffer
                                03AB  1032 ;
                         50 04  AC  D0  03AB  1033          MOVL    P2(AP),R0               ; Get user buffer descriptor address
                                03AF  1034          IFNORD  #8,(R0),10$             ; Check accessibility
                            51  60  3C  03B5  1035          MOVZWL  (R0),R1                 ; Get buffer size
                                0F  13  03B8  1036          BEQL    10$                     ; Br if zero - error
                         50 04  A0  D0  03BA  1037          MOVL    4(R0),R0                ; Get buffer address
                   00000000'GF      16  03BE  1038          JSB     G^EXE$READCHK           ; Check access to buffer
                                03C4  1039                                                  ; (no return on no access)
                            32  51  D1  03C4  1040          CMPL    R1,#CNT_BUFSIZ          ; Is buffer long enough?
                                06  1E  03C7  1041          BGEQU   20$                     ; Br if yes
                            50  14  7D  03C9  1042 10$:     MOVQ    S^#SS$_BADPARAM,R0      ; Set error status
                               FD39  31  03CC  1043          BRW     ABORTIO                 ; Abort the I/O request
                                03CF  1044 ;
                                03CF  1045 ; Move driver maintained counters to caller's buffer
                                03CF  1046 ;
                                18  BB  03CF  1047 20$:     PUSHR   #^M<R3,R4>              ; Save registers
                            57  50  D0  03D1  1048          MOVL    R0,R7                   ; Set address of caller's buffer
                            50  04  D0  03D4  1049          MOVL    #UCB$C_XM_DRVCNT,R0     ; Get number of driver counters
                     51  0120  C5  DE  03D7  1050          MOVAL   UCB$L_XM_DRVCNT(R5),R1  ; Get address of driver counters
                     52  FC98  CF  9E  03DC  1051          MOVAB   CNTTAB,R2               ; Get address of ID table
                            87  82  B0  03E1  1052 30$:     MOVW    (R2)+,(R7)+             ; Set counter ID
                            87  81  D0  03E4  1053          MOVL    (R1)+,(R7)+             ; Set counter value
                               F7  50  F5  03E7  1054          SOBGTR  R0,30$                  ; Loop through all driver counters
```

```
                                03EA  1055 ;
                                03EA  1056 ; Move device maintained counters to caller's buffer
                                03EA  1057 ;
          53   0118 C5    D0   03EA  1058          MOVL    UCB$L_XM_BASETAB(R5),R3 ; Get address of basetable
                  58    D4   03EF  1059 40$:      CLRL    R8                      ; Init bitmask
               59   82    B0   03F1  1060          MOVW    (R2)+,R9                ; Get next counter ID
                    30    13   03F4  1061          BEQL    70$                     ; Br if end of table
               87   59    B0   03F6  1062          MOVW    R9,(R7)+                ; Set next ID in buffer
                    87    94   03F9  1063          CLRB    (R7)+                   ; Clear count
            02 59    0C    E1   03FB  1064          BBC     #NMA$V_CNT_MAP,R9,50$   ; Br if not bitmapped
                    87    B4   03FF  1065          CLRW    (R7)+                   ; Clear bitmap
                                0401  1066
               54   82    9A   0401  1067 50$:      MOVZBL  (R2)+,R4                ; Get next basetable counter offset
                    E9    13   0404  1068          BEQL    40$                     ; Br if none - no more with this ID
               56   82    9A   0406  1069          MOVZBL  (R2)+,R6                ; Get next UCB counter offset
                    0B    E1   0409  1070          BBC     #XM$V_STS_ACTIVE,-      ; Br if basetable not active
                F3 44 A5         040B  1071                  UCB$L_DEVDEPEND(R5),50$ ;
          50   6146   6344   81   040E  1072          ADDB3   (R3)[R4],(R1)[R6],R0    ; Add basetable counter to saved value
                    EB    1B   0414  1073          BLEQU   50$                     ; Br if overflow, etc.
            06 59    0C    E1   0416  1074          BBC     #NMA$V_CNT_MAP,R9,60$   ; Br if not bitmapped
                    58    B6   041A  1075          INCW    R8                      ; Increment bitmask
            FD A7    58    A8   041C  1076          BISW    R8,-3(R7)               ; Set bitmap
            FF A7    50    80   0420  1077 60$:      ADDB    R0,-1(R7)               ; Add to count
                    DB    11   0424  1078          BRB     50$                     ; Loop through all device counters
                                0426  1079
                    18    BA   0426  1080 70$:      POPR    #^M<R3,R4>              ; Restore registers
                                0428  1081 ;
                                0428  1082 ; See if counters are to be "cleared".  The controller has its own copy of
                                0428  1083 ; the counters in its RAM, so the basetable copies can't simply be cleared.
                                0428  1084 ; Instead, a negative of the basetable copies will be saved in the UCB and,
                                0428  1085 ; later when the counts are requested, the UCB copies will be added to the
                                0428  1086 ; basetable copies.
                                0428  1087 ;
       24 20 A3    0A    E1   0428  1088 80$:      BBC     #IO$V_CLR_COUNT,IRP$W_FUNC(R3),110$ ; Br if not clearing counters
             0120 C5    7C   042D  1089          CLRQ    UCB$L_RCVBYTCNT(R5)     ; Clear byte counts
             0128 C5    7C   0431  1090          CLRQ    UCB$L_RCVMSGCNT(R5)     ; Clear message counts
          51 0130 C5    9E   0435  1091          MOVAB   UCB$B_XM_DEVCNT(R5),R1  ; Get address of saved counters
       52 0118 C5    03    C1   043A  1092          ADDL3   #3,UCB$L_XM_BASETAB(R5),R2 ; Get address of basetable counters
             59    08    D0   0440  1093          MOVL    #UCB$C_XM_DEVCNT,R9     ; Set number of counters
                   81    94   0443  1094 90$:      CLRB    (R1)+                   ; Clear saved counter
                   0B    E1   0445  1095          BBC     #XM$V_STS_ACTIVE,-      ; Br if basetable not active
                04 44 A5         0447  1096                  UCB$L_DEVDEPEND(R5),100$;
             FF A1    82    8E   044A  1097          MNEGB   (R2)+,-1(R1)            ; Store negative of basetable counter
                   F2 59    F5   044E  1098 100$:     SOBGTR  R9,90$                  ; Loop through counters
                                0451  1099
          50   32    10    78   0451  1100 110$:     ASHL    #16,#CNT_BUFSIZ,R0      ; Set returned buffer size
               50    01    B0   0455  1101          MOVW    S^#SS$_NORMAL,R0        ; Success return
            00000000'GF    17   0458  1102          JMP     G^EXE$FINISHIOC         ; Post the I/O
                                045E  1103
```

```
                                045E  1105          .SBTTL  STARTIO - Start setmode I/O operation
                                045E  1106 ;++
                                045E  1107 ; STARTIO - Start setmode operation
                                045E  1108 ;
                                045E  1109 ; Functional description:
                                045E  1110 ;
                                045E  1111 ; This routine is entered to process a setmode request.  All setmode
                                045E  1112 ; requests are queued to single-stream them.
                                045E  1113 ;
                                045E  1114 ; For all functions a change in the characteristics is done.
                                045E  1115 ;
                                045E  1116 ; For startup, the action is to request and set up the UNIBUS
                                045E  1117 ; map for the base table and receives. This data is saved
                                045E  1118 ; after allocation in the UCB.  After this the base table and
                                045E  1119 ; receive buffer addresses are passed to the device, thus starting
                                045E  1120 ; the protocol running.
                                045E  1121 ;
                                045E  1122 ; For shutdown, the device is master cleared and all buffers and
                                045E  1123 ; quotas are returned.
                                045E  1124 ;
                                045E  1125 ; Inputs:
                                045E  1126 ;
                                045E  1127 ;       R3 = I/O packet address
                                045E  1128 ;       R5 = UCB address
                                045E  1129 ;
                                045E  1130 ; Outputs:
                                045E  1131 ;
                                045E  1132 ;       R3 and R5 preserved.
                                045E  1133 ;
                                045E  1134 ;       I/O request completed.
                                045E  1135 ;--
                                045E  1136 STARTIO::                                    ; Start I/O routine
                          06    E1  045E  1137          BBC     #IO$V_STARTUP,-         ; Br if not startup request
                       39 20    A3  0460  1138                  IRP$W_FUNC(R3),10$      ;
                                0463  1139 ;
                                0463  1140 ; Startup request
                                0463  1141 ;
                          0B    E1  0463  1142          BBC     #XM$V_STS_ACTIVE,-      ; Br if it is NOT active
                       31 44    A5  0465  1143                  UCB$L_DEVDEPEND(R5),5$  ;
                    50    0C    A3  3C  0468  1144          MOVZWL  IRP$L_PID(R3),R0        ; Get process index from IRP
                51  00000000'  GF  D0  046C  1145          MOVL    G^SCH$GL_PCBVEC,R1      ; Get address of PCB address vector
                    50    6140  D0  0473  1146          MOVL    (R1)[R0],R0             ; Get PCB address
                       60    A0  D1  0477  1147          CMPL    PCB$L_PID(R0),-         ; Still same process?
                          0C    A3  047A  1148                  IRP$L_PID(R3)           ;
                          11    12  047C  1149          BNEQ    3$                      ; Br if not - forget it
                 50     0080  C0  D0  047E  1150          MOVL    PCB$L_JIB(R0),R0        ; Get JIB address
                   51    30    A3  3C  0483  1151          MOVZWL  IRP$W_BOFF(R3),R1       ; Convert quota to longword
                   20 A0       51  C0  0487  1152          ADDL    R1,JIB$L_BYTCNT(R0)     ; Return byte count quota
                   24 A0       51  C0  048B  1153          ADDL    R1,JIB$L_BYTLM(R0)      ; ..and byte limit quota
                      30    A3  B4  048F  1154 3$:       CLRW    IRP$W_BOFF(R3)          ; Reset quota charge
                 50   02C4  8F  3C  0492  1155          MOVZWL  #SS$_DEVACTIVE,R0       ; Device already started
                          1D  11  0497  1156          BRB     40$                     ; Complete the request
                                0499  1157
                        0024  31  0499  1158 5$:       BRW     STARTUP                 ; Start the device
                                049C  1159 ;
                                049C  1160 ; Shutdown request
                                049C  1161 ;
```

```
                07   E1  049C  1162 10$:     BBC     #IO$V_SHUTDOWN,-          ; Br if not shutdown request
        0F 20 A3         049E  1163                  IRP$W_FUNC(R3),20$        ;
    50   0084 8F   3C    04A1  1164          MOVZWL  #SS$_DEVOFFLINE,R0        ; Assume device not started yet
                0B   E1  04A6  1165          BBC     #XM$V_STS_ACTIVE,-        ; Br if not active
        0B 44 A5         04A8  1166                  UCB$L_DEVDEPEND(R5),40$   ;
              08F7   30  04AB  1167          BSBW    SHUTDOWN                  ; Shutdown the device
                03   11  04AE  1168          BRB     30$                       ;
                         04B0  1169 ;
                         04B0  1170 ; Just a change mode request
                         04B0  1171 ;
              03F3   30  04B0  1172 20$:     BSBW    CHANGE_MODE               ; Change mode and characteristics
                         04B3  1173
        50    01   3C    04B3  1174 30$:     MOVZWL  S^#SS$_NORMAL,R0          ; Set success
      51    44 A5   D0   04B6  1175 40$:     MOVL    UCB$L_DEVDEPEND(R5),R1    ; Set device characteristics
                         04BA  1176          REQCOM                            ; Complete the request
                         04C0  1177
```

```
                                1179          .SBTTL  STARTUP - Start up controller
                                1180 ;++
                                1181 ; STARTUP - Start up controller
                                1182 ;
                                1183 ; Functional description:
                                1184 ;
                                1185 ; This routine starts the controller running. The action is to allocate
                                1186 ; the map registers for the base table and receives. Once this is done,
                                1187 ; the unit is master cleared and the base table and mode are set up.
                                1188 ; The receive buffer list is filled and the receives started.
                                1189 ;
                                1190 ;
                                1191 ; Inputs:
                                1192 ;
                                1193 ;       R3 = I/O packet address
                                1194 ;       R5 = UCB address
                                1195 ;
                                1196 ;       IRP$L_MEDIA(R3) =  New mode buffer
                                1197 ;       IRP$L_SVAPTE(R3) = Address of allocated base table.
                                1198 ;       IRP$W_BOFF(R3) = Quota taken from caller.
                                1199 ;
                                1200 ; Outputs:
                                1201 ;
                                1202 ;       Device started and I/O request completed.
                                1203 ;
                                1204 ;       R3,R5 preserved.
                                1205 ;--
                                1206 STARTUP:                                            ; Startup controller
44 A5   18   08   00   F0       1207          INSV    #0,#8,#24,UCB$L_DEVDEPEND(R5)     ; Reset status and error flags
                03DD   30       1208          BSBW    CHANGE_MODE                       ; Set new characteristics
                                1209 ;
                                1210 ; Initialize the buffer and I/O request queue heads
                                1211 ;
     50   08   D0               1212          MOVL    #UCB$C_XM_QUEUES,R0       ; Set number of queue heads
52   0090 C5   9E               1213          MOVAB   UCB$Q_XM_QUEUES(R5),R2    ; Set address of first head
     82   62   9E               1214 10$:     MOVAB   (R2),(R2)+                ; Set forward link
  82   FC A2   9E               1215          MOVAB   -4(R2),(R2)+              ; Set backward link
     F6 50   F5                 1216          SOBGTR  R0,10$                    ; Loop through all queue heads
                                1217 ;
                                1218 ; Initialize the transmit and receive mapping info vectors.
                                1219 ;
     50   0E   D0               1220          MOVL    #MAX_RCV+MAX_XMT,R0       ; Set number receive and transmit slots
                                1221          ASSUME  UCB$L_XM_RCV_MAP+<4*MAX_RCV> EQ UCB$L_XM_XMT_MAP
51   00D0 C5   DE               1222          MOVAL   UCB$L_XM_RCV_MAP(R5),R1   ; Get mapping vector address
     81   01   CE               1223 20$:     MNEGL   #1,(R1)+                  ; Indicate no mapping info
       FA 50   F5               1224          SOBGTR  R0,20$                    ; Loop through all mapping slots
010A C5   07   90               1225          MOVB    #MAX_RCV,UCB$B_XM_RCV_MAX(R5) ; Set maximum concurrent receives
010B C5   07   90               1226          MOVB    #MAX_XMT,UCB$B_XM_XMT_MAX(R5) ; Set maximum concurrent transmits
011C C5   01   CE               1227          MNEGL   #1,UCB$L_XM_BASEMAP(R5) ; Set no mapping for basetable yet
                                1228
     0100 8F   A3               1229          SUBW3   #BASETAB_SIZE,-           ; Compute quota for receive buffers
010C C5   30 A3                 1230                  IRP$W_BOFF(R3),UCB$W_XM_QUOTA(R5)
51   010C C5   3C               1231          MOVZWL  UCB$W_XM_QUOTA(R5),R1     ; Get buffer quota as longword
  50   42 A5   3C               1232          MOVZWL  UCB$W_DEVBUFSIZ(R5),R0    ; Get buffer size as longword
     51   50   C6               1233          DIVL    R0,R1                     ; Compute maximum number of receive
                                1234                                            ;  buffers based on quota
010A C5   51   91               1235          CMPB    R1,UCB$B_XM_RCV_MAX(R5) ; Is number less than maximum?
```

```
                05   1E   0512  1236          BGEQU   30$                             ; Br if not - value ok
          010A C5   51   90   0514  1237          MOVB    R1,UCB$B_XM_RCV_MAX(R5)         ; Else reduce number to quota
                0C A3   D0   0519  1238 30$:      MOVL    IRP$L_PID(R3),-                 ; Save starter's process ID
               0110 C5         051C  1239                  UCB$L_XM_PID(R5)                ;
                               051F  1240 ;
                               051F  1241 ; Save basetable info
                               051F  1242 ;
     54    2C A3   0C   C1   051F  1243          ADDL3   #BAS_T_DATA,IRP$L_SVAPTE(R3),R4 ; Get basetable address
          0118 C5   54   D0   0524  1244          MOVL    R4,UCB$L_XM_BASETAB(R5)         ; Save in UCB
                2C A3   D4   0529  1245          CLRL    IRP$L_SVAPTE(R3)                ; No buffer or quota
                30 A3   B4   052C  1246          CLRW    IRP$W_BOFF(R3)                  ; for I/O post
                   08   A8   052F  1247          BISW    #UCB$M_XM_INITED,-              ; Indicate UCB fields now initialized
                68 A5         0531  1248                  UCB$W_DEVSTS(R5)                ; sufficiently so shutdown can cleanup
                               0533  1249 ;
                               0533  1250 ; Allocate map registers for receive buffers.  The
                               0533  1251 ; unbuffered datapath (DP0) is used for all I/O's due to the fact that
                               0533  1252 ; the controller can initiate retransmissions but on the 11/780, the datapath
                               0533  1253 ; requires purging before it can be reused.
                               0533  1254 ;
                42 A5   B0   0533  1255          MOVW    UCB$W_DEVBUFSIZ(R5),-           ; Set buffer size
                7E A5         0536  1256                  UCB$W_BCNT(R5)
     7C A5    01FF 8F   B0   0538  1257          MOVW    #511,UCB$W_BOFF(R5)             ; Set worst case byte offset
          54    24 A5   D0   053E  1258          MOVL    UCB$L_CRB(R5),R4                ; Get CRB address
                               0542  1259          ASSUME  VEC$W_MAPREG+2 EQ VEC$B_NUMREG
                               0542  1260          ASSUME  VEC$B_NUMREG+1 EQ VEC$B_DATAPATH
                34 A4   D4   0542  1261          CLRL    CRB$L_INTD+VEC$W_MAPREG(R4) ; Clear map register + datapath
               00C0 8F   BB   0545  1262          PUSHR   #^M<R6,R7>                      ; Save regs
          56   00D0 C5   DE   0549  1263          MOVAL   UCB$L_XM_RCV_MAP(R5),R6         ; Get mapping slot address
          57   010A C5   9A   054E  1264          MOVZBL  UCB$B_XM_RCV_MAX(R5),R7         ; Get number of receive slots
           00000000'GF   16   0553  1265 40$:      JSB     G^IOC$ALOUBAMAP                 ; Allocate a set of map registers
                07   50   E9   0559  1266          BLBC    R0,50$                          ; Br if unavailable
            86    34 A4   D0   055C  1267          MOVL    CRB$L_INTD+VEC$W_MAPREG(R4),(R6)+ ; Save map info
                F0 57   F5   0560  1268          SOBGTR  R7,40$                          ; Continue until done
                               0563  1269
               00C0 8F   BA   0563  1270 50$:      POPR    #^M<R6,R7>                      ; Restore regs
                18   50   E9   0567  1271          BLBC    R0,60$                          ; Br if error
                               056A  1272 ;
                               056A  1273 ; Map base table
                               056A  1274 ;
               FE00 8F   AB   056A  1275          BICW3   #^C<VA$M_BYTE>,-                ; Get basetable byte offset
     7C A5    0118 C5         056E  1276                  UCB$L_XM_BASETAB(R5),UCB$W_BOFF(R5)
     7E A5    0100 8F   B0   0573  1277          MOVW    #BASETAB_SIZE,UCB$W_BCNT(R5) ; Set basetable size
           00000000'GF   16   0579  1278          JSB     G^IOC$ALOUBAMAP                 ; Allocate map registers
                08   50   E8   057F  1279          BLBS    R0,70$                          ; Br if allocated
          50    0344 8F   3C   0582  1280 60$:      MOVZWL  #SS$_INSFMAPREG,R0              ; Set insufficient map registers
                   02C4   31   0587  1281          BRW     START_ERROR                     ;
                               058A  1282
                34 A4   D0   058A  1283 70$:      MOVL    CRB$L_INTD+VEC$W_MAPREG(R4),- ; Save basetable mapping info
               011C C5         058D  1284                  UCB$L_XM_BASEMAP(R5)            ;
                   09   EF   0590  1285          EXTZV   S^#VA$V_VPN,-                   ; Get basetable page number
     51    0118 C5   15         0592  1286                  S^#VA$S_VPN,UCB$L_XM_BASETAB(R5),R1
          50   00000000'GF   D0   0597  1287          MOVL    G^MMG$GL_SPTBASE,R0             ; Get SPT address
          78 A5    6041   DE   059E  1288          MOVAL   (R0)[R1],UCB$L_SVAPTE(R5) ; Set PTE address
           00000000'GF   16   05A3  1289          JSB     G^IOC$LOADUBAMAPA               ; Load the basetable map registers
                               05A9  1290          ASSUME  UCB$W_BOFF+2 EQ UCB$W_BCNT
                34 A4   F0   05A9  1291          INSV    CRB$L_INTD+VEC$W_MAPREG(R4),- ; Set BA9-BA15
     7C A5    07   09         05AC  1292                  #9,#7,UCB$W_BOFF(R5)            ;
```

```
                02    07    EF   05B0  1293          EXTZV   #7,#2,-                         ; Get BA16-BA17
             50   34  A4         05B3  1294                  CRB$L_INTD+VEC$W_MAPREG(R4),R0
      7C A5  02   1E   50    F0   05B6  1295          INSV    R0,#30,#2,UCB$W_BOFF(R5) ; Set BA16-BA17
                                 05BC  1296 ;
                                 05BC  1297 ; Master clear the device and notify it of the address of the base table
                                 05BC  1298 ;
                  54    2C B4    D0   05BC  1299 80$:     MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get CSR address
                                 05C0  1300              DSBINT  UCB$B_DIPL(R5)           ; Disable device interrupts
                  03 A4    03    90   05C7  1301              MOVB    #DMC_DMR,XM_O_CSR+1(R4) ; Set DMC/DMR test value
                64    4000 8F    B0   05CB  1302              MOVW    #XM_I_M_MCLR,(R4)        ; Master clear controller
                                 05D0  1303              TIMEWAIT #15,#XM_I_M_RUN,(R4),W ; Wait for RUN - try 150 usecs
                      05 50    E9   05F5  1304              BLBC    R0,85$                   ; Br if device NOT ready
                                 05F8  1305              ENBINT                           ; Else, re-enable interrupts
                         20    11   05FB  1306              BRB     95$                      ; And continue
                                 05FD  1307
                                 05FD  1308 85$:     WFIKPCH 90$,#2                   ; Else, wait about a second for diagnostics
                                 0607  1309 90$:     IOFORK                           ; Schedule a fork process
             64 A5    0040 8F    AA   060D  1310              BICW    #UCB$M_TIMOUT,UCB$W_STS(R5) ; Clear timeout status
                64    8000 8F    B3   0613  1311              BITW    #XM_I_M_RUN,(R4)         ; Device running?
                           03    12   0618  1312              BNEQ    95$                      ; Br if yes
                        022C     31   061A  1313              BRW     START_CTRL_ERROR         ; Else, error
             0C A5    0B84'CF    9E   061D  1314 95$:     MOVAB   W^FORK_PROC,UCB$L_FPC(R5) ; Set Fork process PC address
                   03 A4    03    91   0623  1315              CMPB    #DMC_DMR,XM_O_CSR+1(R4) ; Device a DMC11?
                           03    12   0627  1316              BNEQ    99$                      ; Br if not
                        00B8     31   0629  1317              BRW     120$                     ; Else, must be a DMC11
                   41 A5    02    90   062C  1318 99$:     MOVB    #DT$_DMR11,UCB$B_DEVTYPE(R5) ; Indicate a DMR11
                                 0630  1319 ;
                                 0630  1320 ;       DMR unit - get interface bits, modem signals and configuration bits
                                 0630  1321 ;
                                 0630  1322 ;               Now, get the interface bits (INTMOD, V.35, RS-232, RS-422)
                                 0630  1323 ;
                      64    20   A8   0630  1324              BISW    #XM_I_M_RQI,(R4)         ; Assert RQI
                                 0633  1325              TIMEWAIT #6,#XM_I_M_RDI,(R4),W  ; Wait for controller to come ready
                         17 50    E8   0658  1326              BLBS    R0,105$                  ; Br if port ready
                                 065B  1327              DSBINT  UCB$B_DIPL(R5)           ; Else, disable device interrupts
                                 0662  1328              WFIKPCH 100$,#2                  ; Wait for about 2 seconds
                                 066C  1329 100$:    IOFORK                           ; Create a fork process
                   50    07 A4    90   0672  1330 105$:    MOVB    XM_UCODE+1(R4),R0        ; Get interface bits
             51    50    02    03    EF   0676  1331              EXTZV   #3,#2,R0,R1              ; Get interface bits (INTMOD & V.35)
                   51    51    02    78   067B  1332              ASHL    #MOD$V_XM_INTMOD,R1,R1   ; Shift to start of interface bits
                      47 A5    51    90   067F  1333              MOVB    R1,UCB$L_DEVDEPEND+3(R5) ; Save in UCB @ DEVDEPEND+3
                51    50    FE 8F    78   0683  1334              ASHL    #MOD$V_XM_RS232-6,R0,R1 ; Shift down next two interface bits
                      51    CF 8F    8A   0688  1335              BICB    #^C<MOD$M_XM_RS232!-     ; Remove extraneous bits
                                 068C  1336                      MOD$M_XM_RS422>,R1       ;
                      47 A5    51    88   068C  1337              BISB    R1,UCB$L_DEVDEPEND+3(R5); Save in UCB
                                 0690  1338 ;
                                 0690  1339 ;          Now, get the modem signals
                                 0690  1340 ;
             0150 C5    04 A4    B0   0690  1341              MOVW    XM_PORT(R4),UCB$W_XM_MODSIG(R5) ; Save modem signals
                         64    B4   0696  1342              CLRW    (R4)                     ; Clear RUN, RDI and RQI bits
                                 0698  1343 ;
                                 0698  1344 ;          Now, check the BSEL1 lockout switch - and get the config bits if okay
                                 0698  1345 ;
                64    8000 8F    B3   0698  1346              BITW    #XM_I_M_RUN,(R4)         ; Did we clear RUN?
                           03    13   069D  1347              BEQL    110$                     ; Br if yes - no BSEL1 lockout
                        0115     31   069F  1348              BRW     150$                     ; Else, BSEL1 is locked - skip tests
                      80 8F    88   06A2  1349 110$:    BISB    #MOD$M_XM_BSEL1,-        ; Indicate BSEL1 is ok
```

```
                                                                J 3
                47 A5        06A5  1350                 UCB$L_DEVDEPEND+3(R5)   ;
      06 A4   2296 8F   B0   06A7  1351          MOVW   #UINST_CNF,XM_UCODE(R4) ; Request switch pack bits (config)
         64   0300 8F   A8   06AD  1352          BISW   #XM_I_M_STEPUP!XM_I_M_ROMI,(R4) ; Step microprocessor
                           06B2  1353          WAIT10 #2                      ; Wait 20 useconds
            50   06 A4   B0   06D5  1354          MOVW   XM_UCODE(R4),R0         ; Get configuration bits
   51       50   02    01   EF   06D9  1355          EXTZV  #1,#2,R0,R1             ; Get High Speed & DMC compat mode bits
                           06DE  1356          ASSUME MOD$V_XM_HIGH EQ 0      ;   No shift needed!
            47 A5   51   88   06DE  1357          BISB   R1,UCB$L_DEVDEPEND+3(R5); Save in UCB
                    7A   11   06E2  1358          BRB    130$                    ; Continue in common code
                           06E4  1359 ;
                           06E4  1360 ;       DMC unit
                           06E4  1361 ;
                           06E4  1362 ;          Now, check the BSEL1 lockout - and get configuration if okay
                           06E4  1363 ;
         64   8000 8F   AA   06E4  1364 120$:    BICW   #XM_I_M_RUN,(R4)        ; Clear RUN bit
         64   8000 8F   B3   06E9  1365          BITW   #XM_I_M_RUN,(R4)        ; Did we clear it?
                     03   13   06EE  1366          BEQL   125$                    ; Br if YES - BSEL1 is okay
                   00C4   31   06F0  1367          BRW    150$                    ; Else, BSEL1 is locked out
                           06F3  1368 125$:    ASSUME MOD$V_XM_HIGH EQ 0      ; Else, read rom u-code
                   80 8F   90   06F3  1369          MOVB   #MOD$M_XM_BSEL1,-       ; Indicate BSEL1 is okay
                   47 A5        06F6  1370                 UCB$L_DEVDEPEND+3(R5)   ; ..and assume Low Speed u-code
      06 A4   814D 8F   B0   06F8  1371          MOVW   #UINST_RROM,XM_UCODE(R4) ; Read the DMC rom
         64   0300 8F   A8   06FE  1372          BISW   #XM_I_M_STEPUP!XM_I_M_ROMI,(R4) ; Step the microprocessor
                           0703  1373          WAIT10 #2                      ; Wait 20 useconds
         64   0300 8F   AA   0726  1374          BICW   #XM_I_M_ROMI!XM_I_M_STEPUP,(R4) ; Clear maintenance bits
         64   0400 8F   A8   072B  1375          BISW   #XM_I_M_ROMO,(R4)       ; Set ROMO bit
                           0730  1376          WAIT10 #2                      ; Wait 20 useconds
      06 A4   0390 8F   B1   0753  1377          CMPW   #LS_UCODE,XM_UCODE(R4)  ; Is it low-speed u-code?
                     03   13   0759  1378          BEQL   130$                    ; Br if yes - okay
                           075B  1379          ASSUME MOD$M_XM_HIGH EQ 1
                   47 A5   96   075B  1380          INCB   UCB$L_DEVDEPEND+3(R5)   ; Else, indicate high-speed u-code
                           075E  1381 130$:    DSBINT UCB$B_DIPL(R5)          ; Disable device interrupts
         64   4000 8F   B0   0765  1382          MOVW   #XM_I_M_MCLR,(R4)       ; Master clear controller - again!
                           076A  1383          TIMEWAIT #15,#XM_I_M_RUN,(R4),W ; Wait for RUN - try 150 usecs
                   05 50   E9   078F  1384          BLBC   R0,135$                 ; Br if device NOT ready
                           0792  1385          ENBINT                         ; Else, re-enable interrupts
                     20    11   0795  1386          BRB    150$                    ; And continue
                           0797  1387
                           0797  1388 135$:    WFIKPCH 140$,#2                ; Else, wait about a second
                           07A1  1389 140$:    IOFORK                         ; Schedule a fork process
   64 A5   0040 8F   AA   07A7  1390          BICW   #UCB$M_TIMOUT,UCB$W_STS(R5) ; Clear timeout status
      64   8000 8F   B3   07AD  1391          BITW   #XM_I_M_RUN,(R4)        ; Device running?
                     03   12   07B2  1392          BNEQ   150$                    ; Br if yes
                   0092   31   07B4  1393          BRW    START_CTRL_ERROR        ; Else, error
   0C A5   0884'CF   9E   07B7  1394 150$:    MOVAB  W^FORK_PROC,UCB$L_FPC(R5) ; Set Fork process PC address
                           07BD  1395
                           07BD  1396 ;
                           07BD  1397 ; Set LOOPBACK mode if enabled
                           07BD  1398 ;
                     01    E1   07BD  1399          BBC    #XM$V_CHR_LOOPB,-       ; Br if not loopback mode
                05 44 A5        07BF  1400                 UCB$L_DEVDEPEND(R5),180$;
         64   0800 8F   A8   07C2  1401          BISW   #XM_I_M_LOOPB,(R4)      ; Else, set loopback flag
            50   23    90   07C7  1402 180$:    MOVB   #XM_I_M_RQI!3,R0        ; Set command for basetable-in
                   0097   30   07CA  1403          BSBW   START_REQ_PORT          ; Request port
   04 A4     7C A5   B0   07CD  1404          MOVW   UCB$W_BOFF(R5),XM_PORT(R4) ; Set basetable BA0-BA15
   06 A4     7E A5   B0   07D2  1405          MOVW   UCB$W_BCNT(R5),XM_PORT+2(R4) ; Set basetable BA16-BA17
                           07D7  1406          SETIPL                         ; Disable all interrupts
```

```
                64 A5    05    E0  07DA  1407          BBS     #UCB$V_POWER,UCB$W_STS(R5),- ; Br if power failed
                   6A                07DE  1408                  START_CTRL_ERROR             ;
                64    20    AA    07DF  1409          BICW    #XM_I_M_RQI,(R4)             ; Release port
                                    07E2  1410          SETIPL  UCB$B_FIPL(R5)               ; Restore IPL
                   0081    30       07E6  1411          BSBW    START_WAIT_PORT              ; Wait for controller ready
                                    07E9  1412 ;
                                    07E9  1413 ; Set the device mode and enable interrupts
                                    07E9  1414 ;
                50    21    90       07E9  1415          MOVB    #XM_I_M_RQI!1,R0             ; Set command for control-in
                      76    10       07EC  1416          BSBB    START_REQ_PORT               ; Request port
                   04 A4    B4       07EE  1417          CLRW    XM_PORT(R4)                  ; Clear port (?)
                            AB       07F1  1418          BICW3   #^C<<XM$M_CHR_MOP!-          ; Set mode bits
                                    07F2  1419                      XM$M_CHR_HDPLX!-         ;
                                    07F2  1420                      XM$M_CHR_SLAVE>@8>,-
06 A4   43 A5    F2FF    8F        07F2  1421                  UCB$L_DEVDEPEND-1(R5),XM_PORT+2(R4)
                64    20    8A       07F9  1422          BICB    #XM_I_M_RQI,(R4)             ; Free port
                                    07FC  1423
             014C C5    64    B0       07FC  1424          MOVW    XM_I_CSR(R4),UCB$L_XM_LSTCSR(R5); Save CSR values
          014E C5    02 A4    B0       0801  1425          MOVW    XM_O_CSR(R4),UCB$L_XM_LSTCSR+2(R5)
          0148 C5    04 A4    B0       0807  1426          MOVW    XM_PORT(R4),UCB$L_XM_LSTPRT(R5) ; Save port values
          014A C5    06 A4    B0       080D  1427          MOVW    XM_PORT+2(R4),UCB$L_XM_LSTPRT+2(R5)
          0C A5    0B84'CF    9E       0813  1428          MOVAB   W^FORK_PROC,UCB$L_FPC(R5)    ; Set normal fork process
            02 A4    40 8F    90       0819  1429          MOVB    #XM_O_M_IEO,XM_O_CSR(R4)     ; Enable output interrupts
            02 A4    40 8F    90       081E  1430          MOVB    #XM_O_M_IEO,XM_O_CSR(R4)     ; (again)
                                    0823  1431          SETIPL                               ; Disable all interrupts
                64 A5    05    E0       0826  1432          BBS     #UCB$V_POWER,UCB$W_STS(R5),-  ; Br if power failed
                   1E                082A  1433                  START_CTRL_ERROR             ;
                0800 8F    A8       082B  1434          BISW    #XM$M_STS_ACTIVE,-
                  44 A5               082F  1435                  UCB$L_DEVDEPEND(R5)          ; Set controller now active
                                    0831  1436          SETIPL  UCB$B_FIPL(R5)               ; Restore IPL
                                    0835  1437 ;
                                    0835  1438 ; Start receives and complete the request
                                    0835  1439 ;
                   008E    30       0835  1440          BSBW    FILLRCVLIST                  ; Fill receive buffer list
                0100 8F    A1       0838  1441          ADDW3   #BASETAB_SIZE,-              ; Set quota as bytecount in I/O status
50              010C C5             083C  1442                  UCB$W_XM_QUOTA(R5),R0        ;
50              50    10    78       0840  1443          ASHL    #16,R0,R0                    ; Shift into place
                50    01    B0       0844  1444          MOVW    S^#SS$_NORMAL,R0             ; Set success
                      0D    11       0847  1445          BRB     START_COMPLETE               ;
                                    0849  1446 ;
                                    0849  1447 ; Error during startup - shutdown and complete I/O request
                                    0849  1448 ;
                                    0849  1449 START_CTRL_ERROR:                             ; Controller error during startup
50              0054 8F    3C       0849  1450          MOVZWL  #SS$_CTRLERR,R0              ;
                                    084E  1451 START_ERROR:                                  ; Error during startup
                      50    DD       084E  1452          PUSHL   R0                           ; Save failure status
                   0552    30       0850  1453          BSBW    SHUTDOWN                     ; Shutdown in case partly started
                      50 8EDO       0853  1454          POPL    R0                           ; Restore status
                                    0856  1455 START_COMPLETE:                               ; Complete startup request
51              44 A5    DO       0856  1456          MOVL    UCB$L_DEVDEPEND(R5),R1       ; Get device dependent longword
53              58 A5    DO       085A  1457          MOVL    UCB$L_IRP(R5),R3             ; Get I/O packet address
                                    085E  1458          REQCOM                               ; Complete I/O request
                                    0864  1459
                                    0864  1460 ;++
                                    0864  1461 ; START_REQ_PORT - Startup sequence request port
                                    0864  1462 ; START_WAIT_PORT - Startup sequence wait for port
                                    0864  1463 ;
```

```
                                                     L 3
                     0864  1464 ; Inputs:
                     0864  1465 ;
                     0864  1466 ;       R0 = Command (REQ_PORT only)
                     0864  1467 ;       R4 = CSR address
                     0864  1468 ;       R5 = UCB address
                     0864  1469 ;       00(SP) = Return address
                     0864  1470 ;
                     0864  1471 ; Outputs:
                     0864  1472 ;
                     0864  1473 ;       If unsuccessful, exits to START_CTRL_ERROR.
                     0864  1474 ;--
                     0864  1475 START_REQ_PORT:                          ; Set function and wait
64   50   88           0864  1476         BISB    R0,(R4)                 ; Set command in CSR
64   50   88           0867  1477         BISB    R0,(R4)                 ; (again)
                     086A  1478 START_WAIT_PORT:                         ; Wait for controller ready
                     086A  1479         SETIPL  UCB$B_FIPL(R5)          ; Lower IPL
                     086E  1480         TIMEDWAIT       TIME=#25,-
                     086E  1481                 INS1=<BICB3  #^C<XM_I_M_RDI!XM_I_M_RQI>,(R4),R2>,- ; Get flags
                     086E  1482                 INS2=<BEQL   20$>,-     ; Br if both clear -done
                     086E  1483                 INS3=<CMPB   #XM_I_M_RDI!XM_I_M_RQI,R2>,- ; Check if both set
                     086E  1484                 INS4=<BEQL   20$>,-             ; Br if both set - done
                     086E  1485                 DONELBL=20$
  05 50   E8         0899  1486         BLBS    R0,40$                  ; Br if success
5E   04   C0         089C  1487         ADDL    #4,SP                   ; Else, Pop return address
     A8   11         089F  1488         BRB     START_CTRL_ERROR        ; Exit
                     08A1  1489
                     08A1  1490 40$:    SETIPL  UCB$B_DIPL(R5)          ; Raise IPL again
                  05 08A5  1491         RSB                             ;
                     08A6  1492
```

```
                        08A6  1494          .SBTTL  CHANGE_MODE - Change mode and characteristics
                        08A6  1495 ;++
                        08A6  1496 ; CHANGE_MODE - Change mode and characteristics
                        08A6  1497 ;
                        08A6  1498 ; Functional description:
                        08A6  1499 ;
                        08A6  1500 ; This routine is entered for changing the mode and characteristics on an idle
                        08A6  1501 ; or active unit:
                        08A6  1502 ;
                        08A6  1503 ; Inputs:
                        08A6  1504 ;
                        08A6  1505 ;       R3 = I/O packet address
                        08A6  1506 ;       R5 = UCB address
                        08A6  1507 ;
                        08A6  1508 ;       IRP$L_MEDIA(R3) = Receive buffer size
                        08A6  1509 ;       IRP$L_MEDIA+4(R3) = New device dependent characteristics
                        08A6  1510 ;
                        08A6  1511 ;       The device dependent longword is defined by $XMDEF:
                        08A6  1512 ;
                        08A6  1513 ;       +----------------+---------------+---------------+----------------+
                        08A6  1514 ;       |    not used    |  error status |     status    | characteristics |
                        08A6  1515 ;       +----------------+---------------+---------------+----------------+
                        08A6  1516 ;
                        08A6  1517 ; Outputs:
                        08A6  1518 ;
                        08A6  1519 ;       UCB$W_DEVBUFFSIZ(R5) = Receive buffer size
                        08A6  1520 ;       UCB$L_DEVDEPEND(R5) = Device dependent characteristics
                        08A6  1521 ;--
                        08A6  1522 CHANGE_MODE:
          38 A3    97   08A6  1523          DECB    IRP$L_MEDIA(R3)         ; Valid data buffer?
             1A    12   08A9  1524          BNEQ    10$                     ; Br if not
          3A A3    B0   08AB  1525          MOVW    IRP$L_MEDIA+2(R3),-     ; Set new buffer size
          42 A5         08AE  1526                  UCB$W_DEVBUFSIZ(R5)     ;
    FFFFF7FF 8F    CA   08B0  1527          BICL    #^C<XM$M_STS_ACTIVE>,-  ; Clear all but active flag
          44 A5         08B6  1528                  UCB$L_DEVDEPEND(R5)     ;
    00000800 8F    CA   08B8  1529          BICL    #<XM$M_STS_ACTIVE>,-    ; Clear active flag
          3C A3         08BE  1530                  IRP$L_MEDIA+4(R3)       ;
          3C A3    C8   08C0  1531          BISL    IRP$L_MEDIA+4(R3),-     ; Set new characteristics
          44 A5         08C3  1532                  UCB$L_DEVDEPEND(R5)     ;
                   05   08C5  1533 10$:     RSB
                        08C6  1534
```

```
                        08C6  1536          .SBTTL  FILLRCVLIST - Fill receive buffer list
                        08C6  1537 ;++
                        08C6  1538 ; FILLRCVLIST - Fill receive buffer list
                        08C6  1539 ; ADDRCVLIST - Add a buffer to receive list
                        08C6  1540 ;
                        08C6  1541 ; Functional description:
                        08C6  1542 ;
                        08C6  1543 ; This routine fills the receive buffer free list up to the quota specified
                        08C6  1544 ; at device startup.
                        08C6  1545 ;
                        08C6  1546 ; Inputs:
                        08C6  1547 ;
                        08C6  1548 ;       R2 = Buffer address (ADDRCVLIST only)
                        08C6  1549 ;       R5 = UCB address
                        08C6  1550 ;
                        08C6  1551 ;       IPL = FIPL
                        08C6  1552 ;
                        08C6  1553 ; Outputs:
                        08C6  1554 ;
                        08C6  1555 ;       R5 = UCB address
                        08C6  1556 ;       R1,R2,R4 destroyed.
                        08C6  1557 ;--
                        08C6  1558 FILLRCVLIST:                                 ; Fill receive buffer list
             52   D4    08C6  1559          CLRL    R2                         ; Clear buffer address
             0B   E0    08C8  1560          BBS     #XM$V_STS_ACTIVE,-         ; Continue if device active
      01 44 A5          08CA  1561                  UCB$L_DEVDEPEND(R5),ADDRCVLIST ;
                  05    08CD  1562          RSB                                ;
                        08CE  1563 ADDRCVLIST:                                  ; Add to receive buffer list
                  09    08CE  1564          PUSHR   #^M<R0,R3>                 ; Save registers
                42 A5   08D0  1565 5$:      CMPW    UCB$W_DEVBUFSIZ(R5),-      ; Can new block be allocated ?
                010C C5 08D3  1566                  UCB$W_XM_QUOTA(R5)         ;
                  34    08D6  1567          BGTRU   20$                        ; Br if no - list filled
                  51    08D8  1568          CLRL    R1                         ; Zero size
                004C 8F   08DA  1569          ADDW3   #RCV_T_DATA+CXB$C_TRAILER,- ; Compute needed block size
             51   42 A5   08DE  1570                  UCB$B_DEVBUFSIZ(R5),R1     ;
                  52    D5    08E1  1571          TSTL    R2                     ; Buffer allocated already?
                  09    12    08E3  1572          BNEQ    7$                     ; Br if yes
      00000000'GF         16    08E5  1573          JSB     G^EXE$ALONONPAGED      ; Allocate nonpaged memory
                  17    50    E9    08EB  1574          BLBC    R0,10$             ; Br if failure
               08 A2    51    B0    08EE  1575 7$:      MOVW    R1,RCV_W_BLKSIZE(R2) ; Insert block size
               0A A2    17    90    08F2  1576          MOVB    S^#DYN$C_NET,RCV_B_BLKTYPE(R2) ; Insert block type
            00C0 C5     62    0E    08F6  1577          INSQUE  (R2),UCB$Q_XM_RCV_BUF(R5) ; Insert block on list
                42 A5   A2    08FB  1578          SUBW    UCB$W_DEVBUFSIZ(R5),-  ; Decrement quota
                010C C5       08FE  1579                  UCB$W_XM_QUOTA(R5)     ;
                  52    D4    0901  1580          CLRL    R2                     ; Clear buffer pointer
                  CB    11    0903  1581          BRB     5$                     ;
                        0905  1582 ;
                        0905  1583 10$:     SETBIT  #XM$V_STS_BUFFAIL,-        ; Set buffer alloc failure
                        0905  1584                  UCB$L_DEVDEPEND(R5)        ;
                  10    11    090A  1585          BRB     30$                    ;
                        090C  1586 ;
                        090C  1587 20$:     CLRBIT  #XM$V_STS_BUFFAIL,-        ; Clear buffer alloc failure
                        090C  1588                  UCB$L_DEVDEPEND(R5)        ;
                50 52   D0    0911  1589          MOVL    R2,R0                  ; Set address of buffer
                  06    13    0914  1590          BEQL    30$                    ; Br if none
      00000000'GF         16    0916  1591          JSB     G^COM$DRVDEALMEM       ; Deallocate it
                        091C  1592
```

```
                091C  1593 30$:    DSBINT  UCB$B_DIPL(R5)                  ; Synch access to device
            06  10  0923  1594        BSBB    START_RECEIVE               ; Start the receives
                0925  1595        ENBINT                                  ; Restore IPL
            09  BA  0928  1596        POPR    #^M<R0,R3>                  ; Restore registers
                05  092A  1597 40$:    RSB
                092B  1598
```

```
                                0928  1600          .SBTTL  START_RECEIVE - Start any receives
                                0928  1601 ;++
                                0928  1602 ; START_RECEIVE - Start receives
                                0928  1603 ;
                                0928  1604 ; Functional description:
                                0928  1605 ;
                                0928  1606 ; This routine attempts to start any receives that may be pending.  This
                                0928  1607 ; involves dequeueing a free receive buffer, mapping, and loading its address
                                0928  1608 ; and size into the device.
                                0928  1609 ;
                                0928  1610 ; Inputs:
                                0928  1611 ;
                                0928  1612 ;       R5 = UCB address
                                0928  1613 ;
                                0928  1614 ;       IPL = DIPL
                                0928  1615 ;
                                0928  1616 ; Outputs:
                                0928  1617 ;
                                0928  1618 ;       R5 preserved.
                                0928  1619 ;
                                0928  1620 ;       R0 - R4 destroyed
                                0928  1621 ;--
                                0928  1622 START_RECEIVE:                                    ; Start receive operation
          51   010A C5   9A    0928  1623          MOVZBL  UCB$B_XM_RCV_MAX(R5),R1           ; Get max concurrent receives
51  0108 C5   51   00   EB    0930  1624          FFC     #0,R1,UCB$B_XM_RCV_MAP(R5),R1     ; Get free mapping slot
                   07   13    0937  1625          BEQL    10$                               ; Br if none
          53   00C0 D5   0F    0939  1626          REMQUE  @UCB$Q_XM_RCV_BUF(R5),R3          ; Get a free buffer
                   01   1C    093E  1627          BVC     20$                               ; Br if buffer
                        05    0940  1628 10$:     RSB                                       ;
                              0941  1629 ;
                              0941  1630 ; Mark slot in use and create buffer address / character count image,
                              0941  1631 ; and load UNIBUS adapter map registers.
                              0941  1632 ;
                              0941  1633 20$:     SETBIT  R1,UCB$B_XM_RCV_MAP(R5)           ; Mark slot in use
       0B A3     51    90    0947  1634          MOVB    R1,RCV_B_MAPSLOT(R3)              ; Save mapping slot number used
    54   00D0 C541    DE    094B  1635          MOVAL   UCB$L_XM_RCV_MAP(R5)[R1],R4       ; Get mapping info slot address
       51     48 A3    9E    0951  1636          MOVAB   RCV_T_DATA(R3),R1                 ; Get receive buffer data addr
       0C A3     51    B0    0955  1637          MOVW    R1,RCV_L_BACC(R3)                 ; Set BA0-BA8
                 42 A5    B0    0959  1638          MOVW    UCB$W_DEVBUFSIZ(R5),-             ; Insert character count
                    0E A3          095C  1639                  RCV_L_BACC+2(R3)          ;
0C A3   07   09     64    F0    095E  1640          INSV    (R4),#9,#7,RCV_L_BACC(R3)         ; Set BA9-BA15 from map reg
   50   64   02     07    EF    0964  1641          EXTZV   #7,#2,(R4),R0                     ; Get BA16-BA17 also
0C A3   02   1E     50    F0    0969  1642          INSV    R0,#30,#2,RCV_L_BACC(R3)          ; Set BA16-BA17
                              096F  1643
                   53    DD    096F  1644          PUSHL   R3                                ; Save buffer address
          52   02   A4    3C    0971  1645          MOVZWL  2(R4),R2                          ; Set number of map registers
               53   64    3C    0975  1646          MOVZWL  (R4),R3                           ; Set first map register number
                    54    D4    0978  1647          CLRL    R4                                ; Use unbuffered datapath
          00000000'GF     16    097A  1648          JSB     G^IOC$LOADUBAMAPN                 ; Load the map registers
                   53  8ED0    0980  1649          POPL    R3                                ; Restore buffer address
                   02    10    0983  1650          BSBB    LOAD_PORT                         ; Load buffer into port
                   A4    11    0985  1651          BRB     START_RECEIVE                     ; That was fun - try another
                              0987  1652
```

```
                                        0987  1654          .SBTTL  LOAD_PORT - Load controller input port
                                        0987  1655 ;++
                                        0987  1656 ; LOAD_PORT - Load controller input port
                                        0987  1657 ;
                                        0987  1658 ; Functional description:
                                        0987  1659 ;
                                        0987  1660 ; Request the controller's input port to start an I/O request.  Since the controller
                                        0987  1661 ; doesn't service input requests when it is busy, it may not be attainable
                                        0987  1662 ; in a reasonable amount of time.  In this case, the driver will just have to
                                        0987  1663 ; request an interrupt.
                                        0987  1664 ;
                                        0987  1665 ; Inputs:
                                        0987  1666 ;
                                        0987  1667 ;       R3 = Transmit I/O packet or receive buffer
                                        0987  1668 ;       R5 = UCB address
                                        0987  1669 ;
                                        0987  1670 ;       IPL = DIPL
                                        0987  1671 ;
                                        0987  1672 ; Outputs:
                                        0987  1673 ;
                                        0987  1674 ;       R0 = Success if port loaded immediately
                                        0987  1675 ;       R4 = CSR address
                                        0987  1676 ;       R5 = UCB address
                                        0987  1677 ;
                                        0987  1678 ;       R0-R1 destroyed.
                                        0987  1679 ;--
                                        0987  1680 LOAD_PORT:                              ; Load buffer address/size into port
                                        0987  1681 ;
                                        0987  1682 ; Receive buffers go to head of queue to get initiated first.
                                        0987  1683 ; This prevents the link from shutting down due to receive buffer
                                        0987  1684 ; starvation.
                                        0987  1685 ;
                                        0987  1686 ; Note that receive buffers can go onto queue in any order since, they are
                                        0987  1687 ; merely empty buckets and one is exactly the same as another.  However,
                                        0987  1688 ; transmit buffers contain information and their order must be preserved.
                                        0987  1689 ;
                50    00A4 D5    9E     0987  1690          MOVAB   @UCB$Q_XM_PORT+4(R5),R0 ; Assume request goes at tail of queue
                  0A    0A A3    91     098C  1691          CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP ; Is buffer a transmit?
                        05 13    13     0990  1692          BEQL    10$                     ; Br if yes
                50    00A0 C5    9E     0992  1693          MOVAB   UCB$Q_XM_PORT(R5),R0    ; Else, get address of head of queue
                      60   63    0E     0997  1694 10$:     INSQUE  (R3),(R0)               ; Insert request in queue
                                        099A  1695 ;
                                        099A  1696 LOAD_PORT_ALT:                          ; Entry from PORT_INTR routine, order
                                        099A  1697                                         ; of entries on port queue is preserved
                  54    24 A5    D0     099A  1698          MOVL    UCB$L_CRB(R5),R4        ; Get CRB address
                  54    2C B4    D0     099E  1699          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get CSR address
                        64   20 B3     09A2  1700          BITW    #XM_I_M_RQI,(R4)        ; Is a request already pending ?
                             65 12     09A5  1701          BNEQ    10$                     ; Br if yes - leave
                                        09A7  1702          TIMEWAIT #5,#XM_I_M_RDI,(R4),W,EQL ; Wait for controller to release port
                           3D 50 E9     09CC  1703          BLBC    R0,10$                  ; Br if failure - wait for an interrupt
                        64   20 90     09CF  1704          MOVB    #XM_I_M_RQI,(R4)        ; Request input port
              02 A4   0080 8F B3     09D2  1705          BITW    #XM_O_M_RDO,XM_O_CSR(R4); Is control out pending?
                             28 12     09D8  1706          BNEQ    5$                      ; Br if yes - request interrupt
                                        09DA  1707          TIMEWAIT #5,#XM_I_M_RDI,(R4),W  ; Wait for controller to come ready
                             0D 50 E8     09FF  1708          BLBS    R0,20$                  ; Br if success - port now available
                                        0A02  1709 ;
                                        0A02  1710 ; Port is not currently available - request an interrupt and wait
```

```
                                                E 4
                                0A02  1711 ; until the interrupt occurs.
                                0A02  1712 ;
                64    0060 8F   A8  0A02  1713 5$:     BISW    #XM_I_M_RQI!XM_I_M_IEI,(R4) ; Request interrupt
                64    0060 8F   A8  0A07  1714         BISW    #XM_I_M_RQI!XM_I_M_IEI,(R4) ; (again)
                         50     D4  0A0C  1715 10$:     CLRL    R0                          ; Set failure to load
                         05     05  0A0E  1716         RSB                                 ;
                                0A0F  1717 ;
                                0A0F  1718 ; Port is available - load the buffer address and size into the port
                                0A0F  1719 ;
                53    00A0 D5   0F  0A0F  1720 20$:     REMQUE  @UCB$Q_XM_PORT(R5),R3       ; Get first entry on port queue
                         57     1D  0A14  1721         BVS     INPUT_DONE                  ; Br if none, assume interrupt processed
                                0A16  1722                                                 ;  the request.
                                0A16  1723
                                0A16  1724 LOAD_PORT_AVAIL:                                ; Load port - it's available
                  0A    0A A3   91  0A16  1725         CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP ; Is buffer a transmit?
                         0A     13  0A1A  1726         BEQL    10$                         ; Br if yes
                  17    0A A3   91  0A1C  1727         CMPB    IRP$B_TYPE(R3),S^#DYN$C_NET ; Is buffer a receive buffer?
                         26     13  0A20  1728         BEQL    20$                         ; Br if yes
                                0A22  1729         BUG_CHECK NOBUFPCKT,FATAL                  ; Else, fatal error
                                0A26  1730 ;
                                0A26  1731 ; Load transmit
                                0A26  1732 ;
                00AC D5    63   0E  0A26  1733 10$:     INSQUE  (R3),@UCB$Q_XM_XMT_PND+4(R5)    ; Store on pending queue
              04 A4     38 A3   B0  0A2B  1734         MOVW    IRP$L_MEDIA(R3),XM_PORT(R4)     ; Load buffer address and
              06 A4     3A A3   B0  0A30  1735         MOVW    IRP$L_MEDIA+2(R3),XM_PORT+2(R4) ; character count
00000000'GF    000000FF 8F   C1  0A35  1736         ADDL3   #255,G^EXE$GL_ABSTIM,-             ; Set 255 second timer
                      6C A5        0A40  1737                 UCB$L_DUETIM(R5)
                         03   A8  0A42  1738         BISW    #UCB$M_TIM!UCB$M_INT,-          ; Enable timer
                      64 A5        0A44  1739                 UCB$W_STS(R5)
                         12   11  0A46  1740         BRB     30$                             ;
                                0A48  1741 ;
                                0A48  1742 ; Load receive
                                0A48  1743 ;
                00B4 D5    63   0E  0A48  1744 20$:     INSQUE  (R3),@UCB$Q_XM_RCV_PND+4(R5)    ; Store on pending queue
              04 A4     0C A3   B0  0A4D  1745         MOVW    RCV_L_BACC(R3),XM_PORT(R4)      ; Load buffer address and
              06 A4     0E A3   B0  0A52  1746         MOVW    RCV_L_BACC+2(R3),XM_PORT+2(R4)  ; character count
                64      04   A8  0A57  1747         BISW    #XM_I_M_RCV,(R4)                ; Set receive buffer type
                                0A5A  1748
                                0A5A  1749 30$:     DSBINT                                  ; Disable all interrupts
              05 64 A5    05   E0  0A60  1750         BBS     #UCB$V_POWER,UCB$W_STS(R5),40$  ; Br if powerfailed - forget it
                64    0060 8F   AA  0A65  1751         BICW    #XM_I_M_RQI!XM_I_M_IEI,(R4)     ; Release port, start transfer
                                0A6A  1752 40$:     ENBINT                                  ; Re-enable interrupts
                                0A6D  1753
                                0A6D  1754 INPUT_DONE:
                50      01   3C  0A6D  1755         MOVZWL  S^#SS$_NORMAL,R0                ; Set success loading
                         05     05  0A70  1756         RSB
                                0A71  1757
```

```
                                0A71  1759          .SBTTL  PORT_INTR - Input port ready interrupt service routine
                                0A71  1760 ;++
                                0A71  1761 ; PORT_INTR - Input port ready interrupt service routine
                                0A71  1762 ;
                                0A71  1763 ; Functional description:
                                0A71  1764 ;
                                0A71  1765 ; This interrupt occurs when the port is ready for the driver to pass a
                                0A71  1766 ; buffer address and buffer size to the controller.  Prior to this, a request
                                0A71  1767 ; for the port was made to LOAD_PORT, but the port wasn't available in a
                                0A71  1768 ; short enough amount of time.
                                0A71  1769 ;
                                0A71  1770 ;
                                0A71  1771 ; Inputs:
                                0A71  1772 ;
                                0A71  1773 ;       0(SP) = Address of the unit IDB address
                                0A71  1774 ;       4(SP) - 20(SP) = R1 - R4
                                0A71  1775 ;
                                0A71  1776 ; Outputs:
                                0A71  1777 ;
                                0A71  1778 ;       A receive or transmit is loaded, a check is made for any other
                                0A71  1779 ;       buffers waiting to be loaded and if there are, another request for
                                0A71  1780 ;       the port is made.  Finally, the interrupt is dismissed.
                                0A71  1781 ;
                                0A71  1782 ;       If the interrupt was unexpected, that is no receives or transmits were
                                0A71  1783 ;       pending, the controller is assumed to be in error and is shutdown.
                                0A71  1784 ;--
                                0A71  1785 PORT_INTR:                                   ; Input port ready interrupt
          54    9E    D0      0A71  1786          MOVL    @(SP)+,R4                     ; Get IDB address
       55    18 A4    D0      0A74  1787          MOVL    IDB$L_UCBLST(R4),R5           ; Get UCB address
             0B    E1      0A78  1788          BBC     #XM$V_STS_ACTIVE,-            ; Exit if controller not active
          25 44 A5            0A7A  1789                  UCB$L_DEVDEPEND(R5),INTEXIT
          54    64    D0      0A7D  1790          MOVL    (R4),R4                       ; Get CSR address
50  00A0 8F    64    AB      0A80  1791          BICW3   (R4),#XM_I_M_RDI!XM_I_M_RQI,R0 ; Is a request really pending?
             1A    12      0A86  1792          BNEQ    INTEXIT                       ; Br if not - exit
                           0A88  1793
    53  00A0 D5    0F      0A88  1794          REMQUE  @UCB$Q_XM_PORT(R5),R3         ; Get a waiting buffer/IRP
             1D    1D      0A8D  1795          BVS     INTERR                        ; If VS then none - error
          FF84    30      0A8F  1796          BSBW    LOAD_PORT_AVAIL               ; Load and free the port
                           0A92  1797
    50  00A0 C5    9E      0A92  1798 10$:     MOVAB   UCB$Q_XM_PORT(R5),R0          ; Get address of port queue
          60    50    D1      0A97  1799          CMPL    R0,(R0)                       ; Any more on queue?
             06    13      0A9A  1800          BEQL    INTEXIT                       ; Br if no - exit interrupt
          FEFB    30      0A9C  1801          BSBW    LOAD_PORT_ALT                 ; Attempt to load the port
          F0 50    E8      0A9F  1802          BLBS    R0,10$                        ; Try another
                           0AA2  1803 ;
                           0AA2  1804 ; Exit interrupt
                           0AA2  1805 ;
                           0AA2  1806 INTEXIT:                                     ; Exit interrupt
          50    8E    7D      0AA2  1807          MOVQ    (SP)+,R0                      ; Restore registers
          52    8E    7D      0AA5  1808          MOVQ    (SP)+,R2
          54    8E    7D      0AA8  1809          MOVQ    (SP)+,R4
                02            0AAB  1810          REI
                           0AAC  1811 ;
                           0AAC  1812 ; An unexpected interrupt occured.  Since there is no NOP function to initiate,
                           0AAC  1813 ; the controller must be shutdown.
                           0AAC  1814 ;
                           0AAC  1815 INTERR:                                      ;
```

```
                024F   30  0AAC  1816          BSBW    TIMEOUT                         ; Fake a timeout error
                  F1   11  0AAF  1817          BRB     INTEXIT                         ;
                       0AB1  1818
```

```
                        0AB1  1820          .SBTTL  CONTROL_INTR - Control out interrupt service routine
                        0AB1  1821 ;++
                        0AB1  1822 ; CONTROL_INTR - Control out interrupt service routine
                        0AB1  1823 ;
                        0AB1  1824 ; FUNCTIONAL DESCRIPTION:
                        0AB1  1825 ;
                        0AB1  1826 ; This routine is the control out interupt service routine.  These interrupts
                        0AB1  1827 ; signal receive or transmit buffer done or errors.
                        0AB1  1828 ;
                        0AB1  1829 ; INPUTS:
                        0AB1  1830 ;
                        0AB1  1831 ;       0(SP) = IDB address
                        0AB1  1832 ;       4(SP) - 20(SP) = R1-R5
                        0AB1  1833 ;
                        0AB1  1834 ; OUTPUTS:
                        0AB1  1835 ;
                        0AB1  1836 ; IMPLICIT OUTPUTS:
                        0AB1  1837 ;
                        0AB1  1838 ;       If the interrupt signals an error,
                        0AB1  1839 ;             the port is held and the fork process is scheduled to process
                        0AB1  1840 ;             the error.
                        0AB1  1841 ;
                        0AB1  1842 ;       If the interrupt signals receive done,
                        0AB1  1843 ;             the port is freed;
                        0AB1  1844 ;             the fork process is scheduled to complete any pending I/O;
                        0AB1  1845 ;             the next receive is started if possible.
                        0AB1  1846 ;
                        0AB1  1847 ;       If the interrupt signals transmit done,
                        0AB1  1848 ;             the port is freed;
                        0AB1  1849 ;             the fork process is scheduled to complete the transmit I/O.
                        0AB1  1850 ;--
                        0AB1  1851 CONTROL_INTR:                                ; Control out interrupt
          54    9E    D0  0AB1  1852          MOVL    @(SP)+,R4                 ; Get IDB address
       55    18 A4    D0  0AB4  1853          MOVL    IDB$L_UCBLST(R4),R5       ; Get UCB address
          52    64    D0  0AB8  1854          MOVL    (R4),R2                   ; Get CSR address
                      0B  E1  0ABB  1855          BBC     #XM$V_STS_ACTIVE,-        ; Br if not active
                   E2 44  A5  0ABD  1856                  UCB$L_DEVDEPEND(R5),INTEXIT
          54    02 A2    B0  0AC0  1857          MOVW    XM_O_CSR(R2),R4           ; Get output CSR,
    54    54    10    78  0AC4  1858          ASHL    #16,R4,R4                 ; shift, and
          54    62    B0  0AC8  1859          MOVW    XM_I_CSR(R2),R4           ; get input CSR
       53    06 A2    B0  0ACB  1860          MOVW    XM_PORT+2(R2),R3          ; Get port high word,
    53    53    10    78  0ACF  1861          ASHL    #16,R3,R3                 ; shift, and
       53    04 A2    B0  0AD3  1862          MOVW    XM_PORT(R2),R3            ; get port low word
       04 54    10    E1  0AD7  1863          BBC     #XM_O_V_TYPE+16,R4,10$    ; Br if not error
                   7E    10  0ADB  1864          BSBB    SCHED_FORK                ; Schedule fork process to report error
                   C3    11  0ADD  1865          BRB     INTEXIT                   ;
                         0ADF  1866
 02 A2    0080    8F    AA  0ADF  1867 10$:     BICW    #XM_O_M_RDO,XM_O_CSR(R2); Release output port
53   C0000000    8F    CA  0AE5  1868          BICL    #^XC0000000,R3            ; Clear BA16 and BA17 from BA/CC
                         0AEC  1869                                            ; (not always correct anyway)
       24 54    12    E1  0AEC  1870          BBC     #XM_O_V_RCV+16,R4,40$     ; Br if transmit complete
                         0AF0  1871 ;
                         0AF0  1872 ; Receive completed.  Get the next receive buffer and schedule the fork
                         0AF0  1873 ; process.
                         0AF0  1874 ;
       52    00B0    D5    0F  0AF0  1875          REMQUE  @UCB$Q_XM_RCV_PND(R5),R2 ; Get oldest pending receive
                      B5    1D  0AF5  1876          BVS     INTERR                    ; Error if none
```

```
        0C A2     53   B1  0AF7          CMPW    R3,RCV_L_BACC(R2)         ; Buffer address match?
                  07   13  0AFB          BEQL    30$                       ; Br if yes - ok
     00B0 C5      62   0E  0AFD  20$:    INSQUE  (R2),UCB$Q_XM_RCV_PND(R5) ; Requeue the receive buffer
                  A8   11  0B02          BRB     INTERR                    ; Shutdown the controller
                           0B04
     50    0B A2       9A  0B04  30$:    MOVZBL  RCV_B_MAPSLOT(R2),R0      ; Get mapping slot number used
  EF 0108 C5      50   E5  0B08          BBCC    R0,UCB$B_XM_RCV_MAP(R5),20$; Mark the slot free
        0C A2     53   D0  0B0E          MOVL    R3,RCV_L_BACC(R2)         ; Save byte count
                  1F   11  0B12          BRB     100$                      ;
                           0B14  ;
                           0B14  ; Transmit completed.  Get the next transmit I/O packet and schedule fork
                           0B14  ; process to complete the I/O request.
                           0B14  ;
     52    00A8 D5    0F   0B14  40$:    REMQUE  @UCB$Q_XM_XMT_PND(R5),R2 ; Get pending transmit I/O packet
                  91   1D  0B19          BVS     INTERR                    ; Error if none
                  04   12  0B1B          BNEQ    45$                       ; Br if not last one
                  03   AA  0B1D          BICW    #UCB$M_INT!UCB$M_TIM,-    ; Disable timer
              64  A5       0B1F                  UCB$W_STS(R5)
        38 A2     53   B1  0B21  45$:    CMPW    R3,IRP$L_MEDIA(R2)        ; Buffer address match?
                  08   13  0B25          BEQL    60$                       ; Br if yes - ok
     00A8 C5      62   0E  0B27  50$:    INSQUE  (R2),UCB$Q_XM_XMT_PND(R5) ; Requeue the I/O packet
                FF7D   31  0B2C          BRW     INTERR                    ; Shutdown the controller
                           0B2F
        38 A2     53   D0  0B2F  60$:    MOVL    R3,IRP$L_IOST1(R2)        ; Save byte count
                           0B33
     00BC D5      62   0E  0B33  100$:   INSQUE  (R2),@UCB$Q_XM_POST+4(R5) ; Queue receive buffer or I/O packet
                  03   12  0B38          BNEQ    110$                      ; Br if not first entry
                001E   30  0B3A          BSBW    SCHED_FORK                ; Schedule fork process
                           0B3D  ;
                           0B3D  ; An input buffer may be waiting to be loaded, but for some reason, the
                           0B3D  ; port was unable to be requested.  Check for this condition and if occuring,
                           0B3D  ; attempt to load the port.  Also, since we may have freed-up a receive slot,
                           0B3D  ; it may be possible to load another receive.
                           0B3D  ;
     10 54        05   E0  0B3D  110$:   BBS     #XM_I_V_RQI,R4,120$       ; Br if input request already pending
     50    00A0 C5     9E  0B41  115$:   MOVAB   UCB$Q_XM_PORT(R5),R0      ; Get address of input request queue
        60        50   D1  0B46          CMPL    R0,(R0)                   ; Anything on queue?
                  06   13  0B49          BEQL    120$                      ; Br if no - start receives
                FE4C   30  0B4B          BSBW    LOAD_PORT_ALT             ; Load and free the port
            F0 50      E8  0B4E          BLBS    R0,115$                   ; Br if success - try for another
     03 54        12   E1  0B51  120$:   BBC     #XM_O_V_RCV+16,R4,130$    ; Br if last transfer wasn't receive
                FDD3   30  0B55          BSBW    START_RECEIVE             ; Start any receives
                FF47   31  0B58  130$:   BRW     INTEXIT                   ; Exit
                           0B5B
```

```
                                                J 4
                     0B5B  1922          .SBTTL  SCHED_FORK - Schedule the fork process
                     0B5B  1923 ;++
                     0B5B  1924 ; SCHED_FORK - Schedule the fork process
                     0B5B  1925 ;
                     0B5B  1926 ; Functional description:
                     0B5B  1927 ;
                     0B5B  1928 ; This routine is called to schedule the error and I/O completion fork process.
                     0B5B  1929 ; The last controller port and CSR values are saved for examination.
                     0B5B  1930 ; If the process's execution is already pending, the last port and CSR values
                     0B5B  1931 ; are just saved.
                     0B5B  1932 ;
                     0B5B  1933 ; Inputs:
                     0B5B  1934 ;
                     0B5B  1935 ;       R3 = Last port values
                     0B5B  1936 ;       R4 = Last CSR values
                     0B5B  1937 ;       R5 = UCB address
                     0B5B  1938 ;
                     0B5B  1939 ;       IPL = DIPL or higher
                     0B5B  1940 ;
                     0B5B  1941 ; Outputs:
                     0B5B  1942 ;
                     0B5B  1943 ;       R5 = UCB address
                     0B5B  1944 ;
                     0B5B  1945 ;       UCB$L_XM_LSTPRT(R5) = Last port values
                     0B5B  1946 ;       UCB$L_XM_LSTCSR(R5) = Last CSR values
                     0B5B  1947 ;--
                     0B5B  1948 SCHED_FORK:                                  ; Schedule fork process for execution
           0D   E2   0B5B  1949          BBSS    #UCB$V_XM_FORK_PEND,-      ; Br if fork process scheduling pending
 18 68 A5            0B5D  1950                  UCB$W_DEVSTS(R5),10$
           55   DD   0B60  1951          PUSHL   R5                          ; Save R5
           04   10   0B62  1952          BSBB    5$                          ; Setup fork process
           55 8ED0   0B64  1953          POPL    R5                          ; Restore R5
                05   0B67  1954          RSB                                 ; Return to caller
                     0B68  1955
 55   00000138 8F   C0   0B68  1956 5$:    ADDL    #UCB$B_XM_FKB,R5            ; Point to fork block
           84'AF   9F   0B6F  1957          PUSHAB  B^FORK_PROC                 ; Set address of fork process
      00000000'GF   17   0B72  1958          JMP     G^EXE$FORK                  ; Schedule FORK and return to caller
                     0B78  1959 ;
        05 54   10   E1   0B78  1960 10$:    BBC     #XM_O_V_TYPE+16,R4,20$      ; Br if not an error to handle
      0148 C5   53   7D   0B7C  1961          MOVQ    R3,UCB$L_XM_LSTPRT(R5)      ; Save last port and CSR values
                05   0B81  1962 20$:    RSB                                 ;
                     0B82  1963
```

```
                                0B82  1965          .SBTTL  FORK_PROC - Error and I/O completion fork process
                                0B82  1966 ;++
                                0B82  1967 ; FORK_PROC - Error and I/O completion fork process
                                0B82  1968 ;
                                0B82  1969 ; Functional description:
                                0B82  1970 ;
                                0B82  1971 ; This routine is called as a fork process to handle errors and I/O
                                0B82  1972 ; completions.
                                0B82  1973 ;
                                0B82  1974 ; Inputs:
                                0B82  1975 ;
                                0B82  1976 ;       R3 = Last port values
                                0B82  1977 ;       R4 = Last CSR values
                                0B82  1978 ;       R5 = UCB address at FORK BLOCK
                                0B82  1979 ;
                                0B82  1980 ;       IPL = FIPL
                                0B82  1981 ;
                                0B82  1982 ; Outputs:
                                0B82  1983 ;
                                0B82  1984 ;       R5 preserved.
                                0B82  1985 ;--
                     017C'      0B82  1986          .WORD   TIMEOUT-.                   ; Offset to timeout routine
                                0B84  1987 FORK_PROC:                                   ; Error/completion fork process
                                0B84  1988          CLRBIT  #UCB$V_XM_FORK_PEND,-       ; Clear fork process scheduling pending
                                0B84  1989                  UCB$W_DEVSTS-UCB$B_XM_FKB(R5)
          55    00000138 8F   C2  0B8A  1990          SUBL    #UCB$B_XM_FKB,R5            ; Point to UCB
                03 54    10   E1  0B91  1991          BBC     #XM_O_V_TYPE+16,R4,20$      ; Br if not error
                         0180 30  0B95  1992          BSBW    DEVICE_ERROR                ; Handle the error
                                0B98  1993 ;
                                0B98  1994 ; Complete any transmits or receives
                                0B98  1995 ;
          52    00B8 D5   0F  0B98  1996 20$:    REMQUE  @UCB$Q_XM_POST(R5),R2       ; Get next completed block
                     01   1C  0B9D  1997          BVC     23$                         ; Br if one
                     05   0B9F  1998          RSB                                 ; Else, return
            0A    0A A2   91  0BA0  1999 23$:    CMPB    IRP$B_TYPE(R2),S^#DYN$C_IRP ; Was it a transmit I/O?
                     47   13  0BA4  2000          BEQL    50$                         ; Br if yes - complete it
            17    0A A2   91  0BA6  2001          CMPB    IRP$B_TYPE(R2),S^#DYN$C_NET ; Was it a receive?
                     04   13  0BAA  2002          BEQL    24$                         ; Br if yes
                          0BAC  2003          BUG_CHECK NOBUFPCKT,FATAL           ; Else, fatal error
                          0BB0  2004 ;
                          0BB0  2005 ; Receive completed - if there is a pending receive I/O request, complete it.
                          0BB0  2006 ; Otherwise, queue the buffer and, if enabled, send a message to mailbox.
                          0BB0  2007 ;
            51    0E A2   3C  0BB0  2008 24$:    MOVZWL  RCV_L_BACC+2(R2),R1         ; Get the byte count
                          0BB4  2009          ADDLC   R1,UCB$L_RCVBYTCNT(R5)      ; Update byte count
                0128 C5   D6  0BC0  2010          INCL    UCB$L_RCVMSGCNT(R5)         ; Update message count
          53    0098 D5   0F  0BC4  2011          REMQUE  @UCB$Q_XM_RCV_REQ(R5),R3    ; Remove waiting receive I/O request
                     04   1D  0BC9  2012          BVS     25$                         ; Br if none - queue for later
                     6B   10  0BCB  2013          BSBB    FINISH_RCV_IO               ; Else, finish the I/O
                     C9   11  0BCD  2014          BRB     20$
                          0BCF  2015
          00CC D5   62   0E  0BCF  2016 25$:    INSQUE  (R2),@UCB$Q_XM_RCV_MSG+4(R5); Else, queue message buffer
                     54   D4  0BD4  2017          CLRL    R4                          ; Set no mailbox
                     08   E0  0BD6  2018          BBS     #UCB$V_XM_NOTIF,-           ; Br if already notified
                04 68 A5       0BD8  2019                  UCB$W_DEVSTS(R5),30$        ;
            54    00'8F   9A  0BDB  2020          MOVZBL  #MSG$_XM_DATAVL,R4          ; Set message type
                     00D8 30  0BDF  2021 30$:    BSBW    POKE_USER                   ; Poke the user
```

```
                06 50   E9  0BE2  2022          BLBC    R0,40$                          ; If low clear then not sent
              0800 8F   A8  0BE5  2023          BISW    #UCB$M_XM_NOTIF,-               ; Set notified
                68 A5       0BE9  2024                  UCB$W_DEVSTS(R5)                ;
                   AB   11  0BEB  2025 40$:      BRB     20$                             ;
                            0BED  2026
                            0BED  2027 ;
                            0BED  2028 ; Transmit completed - deallocate the map registers and complete the I/O
                            0BED  2029 ; request.  If there is a transmit request waiting for mapping resources,
                            0BED  2030 ; restart it.
                            0BED  2031 ;
            53      52      D0  0BED  2032 50$:     MOVL    R2,R3                           ; Get I/O packet address
          51    32 A3       3C  0BF0  2033          MOVZWL  IRP$W_BCNT(R3),R1               ; Get byte count
                            0BF4  2034          ADDLC   R1,UCB$L_XMTBYTCNT(R5)          ; Update byte count
            012C C5       D6  0C00  2035          INCL    UCB$L_XMTMSGCNT(R5)             ; Update message count
          51    3C A3       9A  0C04  2036          MOVZBL  IRP$L_MEDIA+4(R3),R1            ; Get mapping slot number used
                            0C08  2037          CLRBIT  R1,UCB$B_XM_XMT_MAP(R5)         ; Clear in use flag
            52    24 A5       D0  0C0E  2038          MOVL    UCB$L_CRB(R5),R2                ; Get CRB address
      34 A2   00EC C541       D0  0C12  2039          MOVL    UCB$L_XM_XMT_MAP(R5)[R1],-      ; Setup map register data
                            0C19  2040                  CRB$L_INTD+VEC$W_MAPREG(R2)
         00EC C541    01    CE  0C19  2041          MNEGL   #1,UCB$L_XM_XMT_MAP(R5)[R1]     ; Set mapping data not allocated
                            0C1F  2042          RELMPR                                  ; Release the map registers
            50    32 A3       B0  0C25  2043          MOVW    IRP$W_BCNT(R3),R0               ; Get count of bytes transmitted
          50    50    10      78  0C29  2044          ASHL    #16,R0,R0                       ; Shift into place
                50    01      B0  0C2D  2045          MOVW    S^#SS$_NORMAL,R0                ; Set success
                      3E      10  0C30  2046          BSBB    IO_DONE                         ; Post the I/O
                            0C32  2047
                    F4DE    30  0C32  2048          BSBW    XMT_START_ALT                   ; Continue any waiting requests
                    FF60    31  0C35  2049 70$:     BRW     20$                             ;
                            0C38  2050
```

```
                                0C38  2052          .SBTTL  FINISH_RCV_IO - Finish receive I/O processing
                                0C38  2053 ;++
                                0C38  2054 ; FINISH_RCV_IO - Finish receive I/O processing
                                0C38  2055 ;
                                0C38  2056 ; FUNCTIONAL DESCRIPTION:
                                0C38  2057 ;
                                0C38  2058 ; This routine completes a receive operation that has been matched with a
                                0C38  2059 ; message block. After the receive has been completed the message free list is
                                0C38  2060 ; filled and a receive is started if needed.
                                0C38  2061 ;
                                0C38  2062 ; INPUTS:
                                0C38  2063 ;
                                0C38  2064 ;       R2 = message buffer address
                                0C38  2065 ;       R3 = I/O packet address
                                0C38  2066 ;       R5 = UCB address
                                0C38  2067 ;
                                0C38  2068 ;       IPL = FIPL
                                0C38  2069 ;
                                0C38  2070 ; OUTPUTS:
                                0C38  2071 ;
                                0C38  2072 ;       R5 = UCB address
                                0C38  2073 ;
                                0C38  2074 ;       The request is completed via I/O post.
                                0C38  2075 ;--
                                0C38  2076 FINISH_RCV_IO:                               ; Finish receive I/O request
        2C A3    52     D0      0C38  2077          MOVL    R2,IRP$L_SVAPTE(R3)        ; Save block address
        62   48 A2     9E      0C3C  2078          MOVAB   RCV_T_DATA(R2),(R2)        ; Set address of received data
    04 A2    38 A3     D0      0C40  2079          MOVL    IRP$L_MEDIA(R3),4(R2)      ; Set address of user buffer
        42   A5        A0      0C45  2080          ADDW    UCB$W_DEVBUFSIZ(R5),-      ; Adjust receive buffer quota
           010C C5              0C48  2081                  UCB$W_XM_QUOTA(R5)         ;
        51    0E A2     B0      0C4B  2082          MOVW    RCV_L_BACC+2(R2),R1        ; Get size of transfer
        32 A3    51     B1      0C4F  2083          CMPW    R1,IRP$W_BCNT(R3)          ; Request larger than actual?
                 04     1B      0C53  2084          BLEQU   20$                        ; Br if no
        51    32 A3     3C      0C55  2085          MOVZWL  IRP$W_BCNT(R3),R1          ; Set size to minimum of two sizes
        32 A3    51     B0      0C59  2086 20$:     MOVW    R1,IRP$W_BCNT(R3)          ; Set size to transfer
     50    51    10     78      0C5D  2087          ASHL    #16,R1,R0                  ; Set up status
                 07     12      0C61  2088          BNEQ    25$                        ; Br if success
     50    0054 8F     B0      0C63  2089          MOVW    #SS$_CTRLERR,R0            ; Set data path error
                 03     11      0C68  2090          BRB     30$                        ;
           50    01     B0      0C6A  2091 25$:     MOVW    S^#SS$_NORMAL,R0           ; Set success
              FC56    30      0C6D  2092 30$:     BSBW    FILLRCVLIST                ; Load another receive
                              0C70  2093 ;
                              0C70  2094 ; Complete a transfer I/O request
                              0C70  2095 ;
                              0C70  2096 IO_DONE:                                     ; Complete a transfer I/O request
        38 A3    50     D0      0C70  2097          MOVL    R0,IRP$L_IOST1(R3)         ; Set status and size
     3C A3    44 A5     D0      0C74  2098          MOVL    UCB$L_DEVDEPEND(R5),IRP$L_IOST2(R3) ; Set other info
     13 2A A3    07     E1      0C79  2099          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),10$ ; Br if no diagnostic buffer
  50    4C B3    08     C1      0C7E  2100          ADDL3   #8,@IRP$L_DIAGBUF(R3),R0 ; Address buffer past start time
80    00000000'GF    7D      0C83  2101          MOVQ    G^EXE$GQ_SYSTIME,(R0)+     ; Insert stop time
   80    0082 C5     3C      0C8A  2102          MOVZWL  UCB$W_ERRCNT(R5),(R0)+     ; Insert error counter
                 06     10      0C8F  2103          BSBB    REGDUMP                    ; Dump registers
        00000000'GF    17      0C91  2104 10$:     JMP     G^COM$POST                 ; Post the I/O and return
                              0C97  2105
```

```
                                N 4
                     0C97  2107          .SBTTL  REGDUMP - Error log and diagnostics register dump
                     0C97  2108 ;++
                     0C97  2109 ; REGDUMP - Error log and diagnostics register dump routine
                     0C97  2110 ;
                     0C97  2111 ; Functional description:
                     0C97  2112 ;
                     0C97  2113 ; This routine is used to return the controller error counters if a diagnostic
                     0C97  2114 ; buffer was specified for an I/O request.
                     0C97  2115 ;
                     0C97  2116 ; Inputs:
                     0C97  2117 ;
                     0C97  2118 ;       R0 = Diagnostic buffer address
                     0C97  2119 ;       R5 = UCB address
                     0C97  2120 ;
                     0C97  2121 ; Outputs:
                     0C97  2122 ;
                     0C97  2123 ;       R5 = UCB address
                     0C97  2124 ;
                     0C97  2125 ;       R0-R1 destroyed.
                     0C97  2126 ;--
                     0C97  2127 REGDUMP:                                ; Dump registers and counters
        80    08   9A  0C97  2128          MOVZBL  #8,(R0)+             ; Insert number longwords returned
  80    014C  C5   D0  0C9A  2129          MOVL    UCB$L_XM_LSTCSR(R5),(R0)+ ; Insert last CSR value
  80    0148  C5   D0  0C9F  2130          MOVL    UCB$L_XM_LSTPRT(R5),(R0)+ ; Insert last port value
              80   7C  0CA4  2131          CLRQ    (R0)+                ; Zero error counters
              0B   E1  0CA6  2132          BBC     #XM$V_STS_ACTIVE,-   ; Br if not active
     0A 44    A5       0CA8  2133                  UCB$L_DEVDEPEND(R5),10$ ;
  51  0118    C5   D0  0CAB  2134          MOVL    UCB$L_XM_BASETAB(R5),R1 ; Get address of base table
                       0CB0  2135          ASSUME  UCB$C_XM_DEVCNT EQ 8
FB A0   03    A1   7D  0CB0  2136          MOVQ    3(R1),-8(R0)         ; Return error counters
              80   7C  0CB5  2137 10$:     CLRQ    (R0)+                ; Clear other counters
              60   7C  0CB7  2138          CLRQ    (R0)                 ;
                   05  0CB9  2139          RSB
                       0CBA  2140
```

```
                                0CBA  2142          .SBTTL  POKE_USER - Poke user process on attention condition
                                0CBA  2143 ;++
                                0CBA  2144 ; POKE_USER - Poke user process on attention condition
                                0CBA  2145 ;
                                0CBA  2146 ; Functional description:
                                0CBA  2147 ;
                                0CBA  2148 ; This routine is used when data is available or a controller error occurs.
                                0CBA  2149 ; the action is to deliver any attention AST's and send a message to the
                                0CBA  2150 ; associated mailbox.
                                0CBA  2151 ;
                                0CBA  2152 ; Inputs:
                                0CBA  2153 ;
                                0CBA  2154 ;       R4 = Mailbox message type
                                0CBA  2155 ;          = Zero if none
                                0CBA  2156 ;       R5 = UCB address
                                0CBA  2157 ;
                                0CBA  2158 ; Outputs:
                                0CBA  2159 ;
                                0CBA  2160 ;       R0 = Low bit clear only if user is not notified
                                0CBA  2161 ;       R5 = UCB address
                                0CBA  2162 ;--
                                0CBA  2163 POKE_USER:                                   ; Poke user process
                          7E    D4  0CBA  2164          CLRL    -(SP)                   ; Assume failure
                          54    DD  0CBC  2165          PUSHL   R4                      ; Save message type
                    0114  C5    9E  0CBE  2166          MOVAB   UCB$L_XM_AST(R5),R1     ; Get AST listhead
                          61    D5  0CC3  2167          TSTL    (R1)                    ; Empty ?
                          18    13  0CC5  2168          BEQL    17$                     ; If so, branch
                       04 AE    D6  0CC7  2169          INCL    4(SP)                   ; Indicate success
                       54   51  D0  0CCA  2170          MOVL    R1,R4                   ; Copy listhead address
                       51   61  D0  0CCD  2171 10$:     MOVL    (R1),R1                 ; Get address of next block
                          07    13  0CD0  2172          BEQL    15$                     ; Br if none - done
                       44 A5    D0  0CD2  2173          MOVL    UCB$L_DEVDEPEND(R5),-   ; Save status as new AST parameter
                       1C A1        0CD5  2174                  ACB$L_KAST+4(R1)        ;
                          F4    11  0CD7  2175          BRB     10$
                00000000'GF     16  0CD9  2176 15$:     JSB     G^COM$DELATTNAST        ; Deliver the AST's
                                0CDF  2177
                          54 8ED0   0CDF  2178 17$:     POPL    R4                      ; Get mailbox message type
                          16    13  0CE2  2179          BEQL    30$                     ; Br if none - no mailbox message
                    53    60 A5    D0  0CE4  2180          MOVL    UCB$L_AMB(R5),R3        ; Get mailbox message address
                          10    13  0CE8  2181          BEQL    30$                     ; Br if none
                0B 44 A5    04    E1  0CEA  2182          BBC     #XM$V_CHR_MBX,UCB$L_DEVDEPEND(R5),30$ ; Br if disabled
                00000000'GF     16  0CEF  2183          JSB     G^EXE$SNDEVMSG          ; Send the mailbox message
                       05 8E    E9  0CF5  2184          BLBC    (SP)+,35$               ; If AST failed, keep R0
                          01    DD  0CF8  2185          PUSHL   #1                      ; Else force success
                          50 8ED0   0CFA  2186 30$:     POPL    R0                      ; Set status
                             05     0CFD  2187 35$:     RSB                             ;
                                0CFE  2188
```

```
                                OCFE  2190          .SBTTL  TIMEOUT - Transmit timeout handler
                                OCFE  2191 ;++
                                OCFE  2192 ; TIMEOUT - Transmit timeout handler
                                OCFE  2193 ;
                                OCFE  2194 ; Functional description:
                                OCFE  2195 ;
                                OCFE  2196 ; This routine is called by the system clock routine to handle a timed-out
                                OCFE  2197 ; unit.  Transmits are the only I/O that is timed for this device.  If it
                                OCFE  2198 ; has timed-out, the error handling fork process is scheduled.
                                OCFE  2199 ;
                                OCFE  2200 ; Inputs:
                                OCFE  2201 ;
                                OCFE  2202 ;       R5 = UCB address
                                OCFE  2203 ;
                                OCFE  2204 ; Outputs:
                                OCFE  2205 ;
                                OCFE  2206 ;       R5 is preserved.
                                OCFE  2207 ;--
                                OCFE  2208 TIMEOUT:                                     ; Timeout handler
                          08    E1  OCFE  2209          BBC     #XM$V_STS_ACTIVE,-      ; Br if controller inactive
                       14 44 A5       0D00  2210                   UCB$L_DEVDEPEND(R5),20$
                    53    01    18   78  0D03  2211          ASHL    #XM_E_V_TIMEOUT+16,#1,R3 ; Set timeout flag
                 04 64 A5    05    E1  0D07  2212          BBC     #UCB$V_POWER,UCB$W_STS(R5),10$  ; Br if not powerfail
                                0D0C  2213          SETBIT  #XM_E_V_POWER+16,R3     ; Set powerfail flag too
                    54    01    10   78  0D10  2214 10$:     ASHL    #XM_D_V_TYPE+16,#1,R4   ; Set error flag
                             FE44    30  0D14  2215          BSBW    SCHED_FORK              ; Schedule the fork process
                                05    0D17  2216 20$:     RSB
                                      0D18  2217
```

```
                                                D 5
                     0D18  2219          .SBTTL  DEVICE_ERROR - Device error handler
                     0D18  2220 ;++
                     0D18  2221 ; DEVICE_ERROR - Device error handler
                     0D18  2222 ;
                     0D18  2223 ; Functional description:
                     0D18  2224 ;
                     0D18  2225 ; This procedure is called to handle device errors.  If the error is non-fatal,
                     0D18  2226 ; the action is simply to, if enabled, send a mailbox message to the device
                     0D18  2227 ; owner.  If the error is fatal, the fatal error status is saved away in
                     0D18  2228 ; the UCB, if enabled, a mailbox message is sent to the device owner, and the
                     0D18  2229 ; device is shutdown.
                     0D18  2230 ;
                     0D18  2231 ; Inputs:
                     0D18  2232 ;
                     0D18  2233 ;       R3 = Last port values
                     0D18  2234 ;       R4 = Last CSR values
                     0D18  2235 ;       R5 = UCB address
                     0D18  2236 ;
                     0D18  2237 ;       IPL = FIPL
                     0D18  2238 ;
                     0D18  2239 ; Outputs:
                     0D18  2240 ;
                     0D18  2241 ;       R5 preserved.
                     0D18  2242 ;--
                     0D18  2243 DEVICE_ERROR:                           ; Device error handler
          50   24 A5   D0  0D18  2244         MOVL    UCB$L_CRB(R5),R0          ; Get CRB address
          50   2C B0   D0  0D1C  2245         MOVL    @CRB$L_INTD+VEC$L_IDB(R0),R0 ; Get CSR address
02 A0    0080 8F   AA  0D20  2246         BICW    #XM_O_M_RDO,XM_O_CSR(R0) ; Free the port
53   53    F0 8F   78  0D26  2247         ASHL    #-16,R3,R3                ; Get last port error value
        0082 C5   B6  0D2B  2248         INCW    UCB$W_ERRCNT(R5)          ; Increment error count
     53   0F98 8F   B3  0D2F  2249         BITW  #<XM_E_M_PROCERR!-           ; Was error a fatal error?
                     0D34  2250                 XM_E_M_NONEXMEM!-
                     0D34  2251                 XM_E_M_START!-
                     0D34  2252                 XM_E_M_LOST!-
                     0D34  2253                 XM_E_M_POWER!-
                     0D34  2254                 XM_E_M_TIMEOUT!-
                     0D34  2255                 XM_E_M_MOP>,R3
              08   12  0D34  2256         BNEQ    20$                       ; Br if yes
        45 A5   53   88  0D36  2257         BISB    R3,UCB$L_DEVDEPEND+1(R5); Save error status
        54    00'8F   9A  0D3A  2258         MOVZBL  #MSG$_XM_ATTN,R4          ; Set mailbox message type
            FF79   31  0D3E  2259         BRW     POKE_USER                 ; If enabled, send mailbox message
                     0D41  2260                                           ; and return
                     0D41  2261 ;
                     0D41  2262 ; Fatal error - device must be shutdown
                     0D41  2263 ;
            0800 8F   AA  0D41  2264 20$:    BICW    #XM$M_STS_ACTIVE,-        ; Clear active flag
              44 A5       0D45  2265                 UCB$L_DEVDEPEND(R5)       ;
                     0D47  2266         ASSUME  <XM_E_M_MOP!XM_E_M_LOST!XM_E_M_START> LE <^XFF>
46 A5   53   67 8F   88  0D47  2267         BICB3   #^C<XM_E_M_MOP!-           ; Save MOP, lost, and start flags
                     0D4D  2268                     XM_E_M_LOST!-             ;
                     0D4D  2269                     XM_E_M_START>,R3,-         ;
                     0D4D  2270                 UCB$L_DEVDEPEND+2(R5)     ;
        14 53   09   E0  0D4D  2271         BBS     #XM_E_V_PROCERR,R3,40$    ; Br if procedure error - don't notify
                     0D51  2272         ASSUME  <XM$M_ERR_FATAL@-16> LE <^XFF>
              01   88  0D51  2273         BISB    #XM$M_ERR_FATAL@-16,-     ; Set fatal error flag
              46 A5       0D53  2274                 UCB$L_DEVDEPEND+2(R5)     ;
        54    00'8F   9A  0D55  2275 30$:     MOVZBL  #MSG$_XM_SHUTDN,R4        ; Set mailbox message type
```

```
                FF5E   30  0D59  2276          BSBW    POKE_USER                       ; If enabled, send mailbox message
              06 50    E8  0D5C  2277          BLBS    R0,40$                          ; Br if successful
            1000 8F    A8  0D5F  2278          BISW    #UCB$M_XM_LOSTERR,-             ; Else, remember lost error
              68 A5        0D63  2279                  UCB$W_DEVSTS(R5)                ;
                 3E    11  0D65  2280 40$:     BRB     SHUTDOWN                        ; Shutdown device and return
                           0D67  2281
```

```
                        0D67  2283          .SBTTL SHUTDOWN - Shut down device
                        0D67  2284          .SBTTL CANCEL - Cancel I/O and Deassign Routine
                        0D67  2285 ;++
                        0D67  2286 ; SHUTDOWN - Shut down device
                        0D67  2287 ; CANCEL - Cancel I/O and Deassign Routine
                        0D67  2288 ;
                        0D67  2289 ; Functional description:
                        0D67  2290 ;
                        0D67  2291 ; This routine is used to shut down the device unit as a result of a
                        0D67  2292 ; SETMODE and SHUTDOWN request, a $CANCEL, or a fatal error. The action is
                        0D67  2293 ; to halt the device, deallocate the basetable, deallocate receive
                        0D67  2294 ; buffers, deallocate all map registers, abort all transmit and receive
                        0D67  2295 ; I/O requests, and restore the quotas to the starting process.
                        0D67  2296 ;
                        0D67  2297 ; Inputs:
                        0D67  2298 ;
                        0D67  2299 ;       R5 = UCB address
                        0D67  2300 ;       R8 = Cancel reason code (zero if $CANCEL else $DASSGN)
                        0D67  2301 ;
                        0D67  2302 ;       IPL = FIPL
                        0D67  2303 ;
                        0D67  2304 ; Outputs:
                        0D67  2305 ;
                        0D67  2306 ;       R0-R3 are destroyed.
                        0D67  2307 ;--
                        0D67  2308
                        0D67  2309 CANCEL:                                      ; Cancel I/O routine
5C A5   B5              0D67  2310          TSTW    UCB$W_REFC(R5)              ; Is this the last $DASSGN or $CANCEL?
   31   13              0D6A  2311          BEQL    100$                        ; Br if yes
                        0D6C  2312 ;
                        0D6C  2313 ; NOT the last $CANCEL or last $DASSGN
                        0D6C  2314 ;
                        0D6C  2315 ; Perform only a selective $CANCEL (same for $DASSGN)
                        0D6C  2316 ;
        03   E1        0D6C  2317          BBC     #UCB$V_XM_INITED,-          ; Br if unit NOT inited
    2B 68 A5            0D6E  2318                  UCB$W_DEVSTS(R5),10$        ;
                        0D71  2319 ;
                        0D71  2320 ; Flush all attention ASTs for this CHANNEL
                        0D71  2321 ;
    00D4 8F   BB        0D71  2322          PUSHR   #^M<R2,R4,R6,R7>            ; Save registers
57  0114 C5   9E        0D75  2323          MOVAB   UCB$L_XM_AST(R5),R7         ; Get address of AST listhead
   56   52    3C        0D7A  2324          MOVZWL  R2,R6                       ; Get channel number
 00000000'GF   16        0D7D  2325          JSB     G^COM$FLUSHATTNS            ; Flush all AST for this channel
    00D4 8F   BA        0D83  2326          POPR    #^M<R2,R4,R6,R7>            ; Restore registers
                        0D87  2327 ;
                        0D87  2328 ; Complete all associated receive IRPs
                        0D87  2329 ;
        56   DD         0D87  2330          PUSHL   R6                          ; Save R6
56  0098 C5   9E        0D89  2331          MOVAB   UCB$Q_XM_RCV_REQ(R5),R6     ; Get address of receive IRPs
      0153   30         0D8E  2332          BSBW    DO_CANCEL                   ; Do the cancel
56  0090 C5   9E        0D91  2333          MOVAB   UCB$Q_XM_XMT_REQ(R5),R6     ; Get address of XMIT IRPs
      014B   30         0D96  2334          BSBW    DO_CANCEL                   ; Do the cancel
        56 8ED0         0D99  2335          POPL    R6                          ; Restore R6
           05           0D9C  2336 10$:     RSB                                 ; Return to caller
                        0D9D  2337
                        0D9D  2338 ;
                        0D9D  2339 ; Last $CANCEL or last $DASSGN request
```

```
                        0D9D  2340 ;
      51   01   9A      0D9D  2341 100$:   MOVZBL  #1,R1                    ; Assume last $DASSGN system service
                        0DA0  2342                                          ; modem is cleared only on last $DASSGN
      58   01   91      0DA0  2343         CMPB    #CAN$C_DASSGN,R8         ; Is this a $DASSGN?
           02   13      0DA3  2344         BEQL    SHUTDOWN_ALT             ; Br if yes, shutdown the modem
                        0DA5  2345
                        0DA5  2346 ;
                        0DA5  2347 ; Shutdown request on unit
                        0DA5  2348 ;
                        0DA5  2349 SHUTDOWN:                                ; Shut down unit
           51   D4      0DA5  2350         CLRL    R1                       ; Do not shutdown the modem
                        0DA7  2351
                        0DA7  2352 SHUTDOWN_ALT:
           04   E1      0DA7  2353         BBC     #UCB$V_ONLINE,-          ; Br if not online
 0B 64     A5           0DA9  2354                 UCB$W_STS(R5),10$        ;
           03   E0      0DAC  2355         BBS     #UCB$V_INITED,-          ; Br if UCB initialized
 07 68     A5           0DAE  2356                 UCB$W_DEVSTS(R5),15$     ;
      03   51   E9      0DB1  2357         BLBC    R1,10$                   ; Br if not to clear DTR
             0161  30    0DB4  2358         BSBW    DISABLE_MODEM            ; Else, disable the modem
                   05    0DB7  2359 10$:    RSB                              ; Exit
                         0DB8  2360 ;
                         0DB8  2361 ; Clear device and device status
                         0DB8  2362 ;
    00D0   8F      BB    0DB8  2363 15$:    PUSHR   #^M<R4,R6,R7>            ; Save registers
      54   24   A5      0DBC  2364         MOVL    UCB$L_CRB(R5),R4         ; Get CRB address
                         0DC0  2365         ASSUME  IDB$L_CSR EQ 0
      54   2C   B4      0DC0  2366         MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get CSR address
                         0DC4  2367         DSBINT  UCB$B_DIPL(R5)           ; Synch access to status flags
  64  4000   8F      B0  0DCB  2368         MOVW    #XM_IM_MCLR,(R4)         ; Master clear the unit
    64 A5      23    AA  0DD0  2369         BICW    #UCB$M_INT!UCB$M_TIM!-   ; Reset device status flags
                         0DD4  2370                 UCB$M_POWER,UCB$W_STS(R5) ;
    0800   8F      AA    0DD4  2371         BICW    #XM$M_STS_ACTIVE,-       ; Reset active flag
        44 A5            0DD8  2372                 UCB$L_DEVDEPEND(R5)      ;
                   AA    0DDA  2373         BICW    #^C<UCB$M_XM_LOSTERR! -  ; Clear all but lost error bit,
                         0DDB  2374                     UCB$M_XM_FORK_PEND>,-; and fork process pending
 68 A5      CFFF  8F      0DDB  2375                 UCB$W_DEVSTS(R5)         ;
           03   51   E9    0DE0  2376         BLBC    R1,17$                   ; Br if not to clear DTR
                 0132  30  0DE3  2377         BSBW    DISABLE_MODEM            ; Disable the modem
                         0DE6  2378 17$:    ENBINT                           ; Restore IPL
                         0DE9  2379 ;
                         0DE9  2380 ; Deallocate all the attention AST control blocks
                         0DE9  2381 ;
      57   0114 C5   9E    0DE9  2382 20$:    MOVAB   UCB$L_XM_AST(R5),R7      ; Get address of AST listhead
           50   67   D0    0DEE  2383         MOVL    (R7),R0                  ; Anything in the list?
                     1B  13  0DF1  2384         BEQL    25$                      ; Br if not
             56   22   A0  3C  0DF3  2385         MOVZWL  ACB$L_KAST+10(R0),R6     ; Force channel match
             52   24   A0  3C  0DF7  2386         MOVZWL  ACB$L_KAST+12(R0),R2     ; Get process index
 54   00000000'GF      D0    0DFB  2387         MOVL    G^SCH$GL_PCBVEC,R4       ; Get PCB address vector address
               54   6442  D0    0E02  2388         MOVL    (R4)[R2],R4              ; Get PCB address
           00000000'GF      16  0E06  2389         JSB     G^COM$FLUSHATTNS         ; Flush AST
                     DB  11    0E0C  2390         BRB     20$                      ; Continue until all flushed
                         0E0E  2391 ;
                         0E0E  2392 ; Release the base table map registers, save the error counters, and
                         0E0E  2393 ; deallocate the base table.
                         0E0E  2394 ;
      54   24   A5      D0    0E0E  2395 25$:    MOVL    UCB$L_CRB(R5),R4         ; Get CRB address
           011C C5      D0    0E12  2396         MOVL    UCB$L_XM_BASEMAP(R5),-   ; Set mapping info
```

```
                    34 A4          0E16  2397                 CRB$L_INTD+VEC$W_MAPREG(R4)
                       0B    19    0E18  2398          BLSS   27$                          ; Br if none
                                   0E1A  2399          RELMPR                             ; Release the map registers
              011C C5  01    CE    0E20  2400          MNEGL  #1,UCB$L_XM_BASEMAP(R5)     ; Reset mapping info
                                   0E25  2401
              50  0118 C5    D0    0E25  2402 27$:     MOVL   UCB$L_XM_BASETAB(R5),R0     ; Get address of base table
                       27    13    0E2A  2403          BEQL   30$                         ; Br if none
              51  50   03    C1    0E2C  2404          ADDL3  #3,R0,R1                    ; Set address of error counters
                  52   08    D0    0E30  2405          MOVL   #UCB$C_XM_DEVCNT,R2         ; Set number of counters
              53  0130 C5    9E    0E33  2406          MOVAB  UCB$B_XM_DEVCNT(R5),R3      ; Get address of saved counters
                  83   81    80    0E38  2407 28$:     ADDB   (R1)+,(R3)+                 ; Add counter to saved counter
                    FA 52    F5    0E3B  2408          SOBGTR R2,28$                      ; Loop through counters
                                   0E3E  2409
                  0118 C5    D4    0E3E  2410          CLRL   UCB$L_XM_BASETAB(R5)        ; Reset state to no table
                50   F4    A0    9E    0E42  2411          MOVAB  -BAS_T_DATA(R0),R0          ; Reset pointer to start of block
          010C C5   0100 8F    A0    0E46  2412          ADDW   #BASETAB_SIZE,UCB$W_XM_QUOTA(R5); Restore quota
           00000000'GF    16    0E4D  2413          JSB    G^COM$DRVDEALMEM            ; Deallocate the base table
                                   0E53  2414 ;
                                   0E53  2415 ; Release the receive and transmit buffer map registers
                                   0E53  2416 ;
                       57    D4    0E53  2417 30$:     CLRL   R7                          ; Init slot number
                                   0E55  2418          ASSUME UCB$L_XM_RCV_MAP+<4*MAX_RCV> EQ UCB$L_XM_XMT_MAP
              56   00D0 C5    9E    0E55  2419          MOVAB  UCB$L_XM_RCV_MAP(R5),R6     ; Get address of mapping slots
                34 A4      86    D0    0E5A  2420 50$:     MOVL   (R6)+,CRB$L_INTD+VEC$W_MAPREG(R4) ; Set mapping info
                       0A    19    0E5E  2421          BLSS   60$                         ; Br if none allocated
                                   0E60  2422          RELMPR                             ; Release the map registers
                  FC A6    01    CE    0E66  2423          MNEGL  #1,-4(R6)                   ; Reset mapping info
                                   0E6A  2424 60$:     CLRBIT R7,UCB$B_XM_RCV_MAP(R5)     ; Clear mapping slot flag
                  E6 57    0E    F2    0E70  2425          AOBLSS #MAX_RCV+MAX_XMT,R7,50$ ; Loop through all mapping slots
                                   0E74  2426 ;
                                   0E74  2427 ; Deallocate all receive buffers and abort all I/O requests
                                   0E74  2428 ;
          56   0090 C5    9E    0E74  2429 90$:     MOVAB  UCB$Q_XM_QUEUES(R5),R6     ; Get address of first queue listhead
               57   08    D0    0E79  2430          MOVL   #UCB$C_XM_QUEUES,R7         ; Get number of queues
            50   00 B6    0F    0E7C  2431 95$:     REMQUE @(R6),R0                    ; Get next I/O packet/buffer
                       29    1D    0E80  2432          BVS    110$                        ; Br if none - queue empty
            0A   0A A0    91    0E82  2433          CMPB   IRP$B_TYPE(R0),S^#DYN$C_IRP ; Is it an I/O packet?
                       0A    13    0E86  2434          BEQL   97$                         ; Br if yes
            17   0A A0    91    0E88  2435          CMPB   IRP$B_TYPE(R0),S^#DYN$C_NET ; Is it a receive buffer?
                       0F    13    0E8C  2436          BEQL   100$                        ; Br if yes
                                   0E8E  2437          BUG_CHECK NOBUFPCKT,FATAL          ; Else, fatal error
                                   0E92  2438
                  53   50    D0    0E92  2439 97$:     MOVL   R0,R3                       ; Set I/O packet address
                  50   2C    3C    0E95  2440          MOVZWL #SS$_ABORT,R0               ; Set I/O status
                     FDD5    30    0E98  2441          BSBW   IO_DONE                     ; Abort the I/O request
                       DF    11    0E9B  2442          BRB    95$                         ;
                                   0E9D  2443
                    42 A5    A0    0E9D  2444 100$:    ADDW   UCB$W_DEVBUFSIZ(R5),-      ; Restore quota
                  010C C5                0EA0  2445                 UCB$W_XM_QUOTA(R5)          ;
           00000000'GF    16    0EA3  2446          JSB    G^COM$DRVDEALMEM            ; Deallocate the receive buffer
                       D1    11    0EA9  2447          BRB    95$                         ;
                                   0EAB  2448
                  56   08    C0    0EAB  2449 110$:    ADDL   #8,R6                       ; Increment queue listhead pointer
                    CB 57    F5    0EAE  2450          SOBGTR R7,95$                      ; Loop through queues
                                   0EB1  2451 ;
                                   0EB1  2452 ; Restore the buffered I/O quota to the starter
                                   0EB1  2453 ;
```

```
      50   0110 C5   3C  0EB1          MOVZWL  UCB$L_XM_PID(R5),R0        ; Get process index of last starter
51  00000000'GF      D0  0EB6          MOVL    G^SCH$GL_PCBVEC,R1         ; Get address of PCB address vector
         50   6140   D0  0EBD          MOVL    (R1)[R0],R0                ; Get PCB address of starter
           60 A0     D1  0EC1          CMPL    PCB$L_PID(R0),-            ; Still same process?
         0110 C5         0EC4                  UCB$L_XM_PID(R5)           ;
              16     12  0EC7          BNEQ    140$                       ; Br if not - forget it
      50   0080 C0   D0  0EC9          MOVL    PCB$L_JIB(R0),R0           ; Get JIB address
      51   010C C5   3C  0ECE          MOVZWL  UCB$W_XM_QUOTA(R5),R1      ; Convert to longword
        20 A0    51  C0  0ED3          ADDL    R1,JIB$L_BYTCNT(R0)        ; Return byte count quota
        24 A0    51  C0  0ED7          ADDL    R1,JIB$L_BYTLM(R0)         ; ..and byte limit quota
         010C C5     B4  0EDB          CLRW    UCB$W_XM_QUOTA(R5)         ; Reset quota
         00D0 8F     BA  0EDF  140$:   POPR    #^M<R4,R6,R7>              ; Restore registers
                     05  0EE3          RSB                                ;
                         0EE4
                         0EE4  DO_CANCEL:                                 ; Cancel the I/O
           53    56  D0  0EE4          MOVL    R6,R3                      ; Copy listhead address
           53    63  D0  0EE7  10$:    MOVL    (R3),R3                    ; Get next entry
           56    53  D1  0EEA          CMPL    R3,R6                      ; At start of list?
                 0F  13  0EED          BEQL    20$                        ; Br if yes
                 0E  10  0EEF          BSBB    CHECK_PKT                  ; Check for match
                 F4  12  0EF1          BNEQ    10$                        ; Br if no match
           53    63  0F  0EF3          REMQUE  (R3),R3                    ; Remove IRP from list
           50    2C  9A  0EF6          MOVZBL  S^#SS$_ABORT,R0            ; Else, set the I/O status return
               FD74  30  0EF9          BSBW    IO_DONE                    ; Abort the I/O request
                 E6  11  0EFC          BRB     DO_CANCEL                  ; Continue from start of list - again
                     05  0EFE  20$:    RSB                                ; Return to caller
                         0EFF
                         0EFF  CHECK_PKT:
        28 A3    52  B1  0EFF          CMPW    R2,IRP$W_CHAN(R3)          ; Channel match?
                 12  12  0F03          BNEQ    20$                        ; Br if no
              0C A3  D5  0F05          TSTL    IRP$L_PID(R3)              ; Is this an Internal IRP?
                 08  14  0F08          BGTR    10$                        ; Br if NO - PID must match
 0110 C5      60 A4  D1  0F0A          CMPL    PCB$L_PID(R4),UCB$L_XM_PID(R5) ; Starter's PID?
                 05  11  0F10          BRB     20$                        ; Done
   0C A3      60 A4  D1  0F12  10$:    CMPL    PCB$L_PID(R4),IRP$L_PID(R3) ; PIDs match?
                     05  0F17  20$:    RSB
                         0F18
```

```
                         0F18          .SBTTL  DISABLE_MODEM - DISABLE THE MODEM LINE DTR
                         0F18   ;++
                         0F18   ; DISABLE_MODEM - DISABLE THE MODEM
                         0F18   ;
                         0F18   ; Functional description:
                         0F18   ;
                         0F18   ;       This routine will clear the DTR line to the modem to hang up
                         0F18   ;       any phone connection still active.
                         0F18   ;
                         0F18   ;
                         0F18   ; Inputs:
                         0F18   ;
                         0F18   ;       R5 = UCB ADDRESS
                         0F18   ;
                         0F18   ; Outputs:
                         0F18   ;
                         0F18   ;       NONE.
                         0F18   ;
                         0F18   ;--
                         0F18
                         0F18   DISABLE_MODEM:                                  ; Disable the modem line (DTR)
                 51   DD 0F18          PUSHL   R1                              ; Save R1
       51     24 A5   D0 0F1A          MOVL    UCB$L_CRB(R5),R1                ; Get CRB address
                         0F1E          ASSUME  IDB$L_CSR EQ 0
       51     2C B1   D0 0F1E          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R1 ; Get CSR address
     61     4000 8F   B0 0F22          MOVW    #XM_I_M_MCLR,(R1)               ; Master clear the unit
  06 A1     A40B 8F   B0 0F27          MOVW    #DROP_DTR,XM_PORT+2(R1) ; Load micro-instruction to drop DTR
  01 A1       82 8F   90 0F2D          MOVB    #EXECUTE_UC,1(R1)               ; Tell controller to execute instruction
                 51 8ED0 0F32          POPL    R1                              ; Restore R1
                      05 0F35          RSB                                     ; Return to caller
                         0F36
                         0F36
                         0F36   XM_END:
                         0F36          .END
```

```
$$$                     =  00000020  R      02
$$$TYP                  =  00000406
$$$WID                  =  00002000
$$OP                    =  00000002
ABORTIO                    00000108  R      03
ACB$L_KAST              =  00000018
ADDRCVLIST                 000008CE  R      03
ALTFDT                     00000208  R      03
ATS_UBA                 =  00000001
BASETAB_SIZE            =  00000100
BAS_B_SPARE                0000000B
BAS_B_TYPE                 0000000A
BAS_C_HEADER               0000000C
BAS_Q_SPARE                00000000
BAS_T_DATA                 0000000C
BAS_W_SIZE                 00000008
BUG$_NOBUFPCKT             ********    X  03
CANS_DASSGN             =  00000001
CANCEL                     00000D67  R      03
CHANGE_MODE                000008A6  R      03
CHECK_PKT                  00000EFF  R      03
CNTTAB                     00000078  R      03
CNT_BUFSIZ              =  00000032
COM$DELATTNAST             ********    X  03
COM$DRVDEALMEM             ********    X  03
COM$FLUSHATTNS             ********    X  03
COM$POST                   ********    X  03
COM$SETATTNAST             ********    X  03
CONTROL_INTR               00000AB1  R      03
CRB$L_INTD              =  00000024
CRB$L_INTD2             =  00000048
CXB$C_HEADER            =  00000048
CXB$C_TRAILER           =  00000004
DC$_SCOM                =  00000020
DDB$L_DDT               =  0000000C
DEV$M_AVL                  ********    X  02
DEV$M_IDV                  ********    X  02
DEV$M_NET                  ********    X  02
DEV$M_ODV                  ********    X  02
DEVICE_ERROR               00000D18  R      03
DISABLE_MODEM              00000F18  R      03
DMC_DMR                 =  00000003
DO_CANCEL                  00000EE4  R      03
DPT$C_LENGTH            =  00000038
DPT$C_VERSION           =  00000004
DPT$INITAB                 00000038  R      02
DPT$REINITAB               00000054  R      02
DPT$TAB                    00000000  R      02
DROP_DTR                =  0000A40B
DT$_DMC11               =  00000001
DT$_DMR11               =  00000002
DYN$C_CRB               =  00000005
DYN$C_DDB               =  00000006
DYN$C_DPT               =  0000001E
DYN$C_IRP               =  0000000A
DYN$C_NET               =  00000017
DYN$C_TQE               =  0000000F
DYN$C_UCB               =  00000010
EXE$ABORTIO                ********    X  03
EXE$ALLOCBUF               ********    X  03
EXE$ALONONPAGED            ********    X  03
EXE$BUFQUOPRC              ********    X  03
EXE$FINISHIO               ********    X  03
EXE$FINISHIOC              ********    X  03
EXE$FORK                   ********    X  03
EXE$GL_ABSTIM              ********    X  03
EXE$GL_TENUSEC             ********    X  03
EXE$GL_UBDELAY             ********    X  03
EXE$GQ_SYSTIME             ********    X  03
EXE$INSTIMQ                ********    X  03
EXE$IOFORK                 ********    X  03
EXE$QIODRVPKT              ********    X  03
EXE$QIORETURN              ********    X  03
EXE$READCHK                ********    X  03
EXE$SNDEVMSG               ********    X  03
EXE$WRITELOCK              ********    X  03
EXECUTE_UC              =  00000082
FILLRCVLIST                000008C6  R      03
FINISH_RCV_IO              00000C38  R      03
FKB$B_FIPL              =  00000008
FKB$C_LENGTH            =  00000018
FKB$L_FR3               =  00000010
FORK_PROC                  00000B84  R      03
FUNCTABLE                  00000038  R      03
FUNCTAB_LEN             =  00000040
IDB$L_CSR               =  00000000
IDB$L_UCBLST            =  00000018
INPUT_DONE                 00000A6D  R      03
INTERR                     00000AAC  R      03
INTEXIT                    00000AA2  R      03
IO$M_CTRL               =  00000200
IO$M_SHUTDOWN           =  00000080
IO$M_STARTUP            =  00000040
IO$V_ATTNAST            =  00000008
IO$V_CLR_COUNT          =  0000000A
IO$V_CTRL               =  00000009
IO$V_DSABLMBX           =  0000000A
IO$V_ENABLMBX           =  00000007
IO$V_NOW                =  00000006
IO$V_RD_COUNT           =  00000008
IO$V_SHUTDOWN           =  00000007
IO$V_STARTUP            =  00000006
IO$_READLBLK            =  00000021
IO$_READPBLK            =  0000000C
IO$_READVBLK            =  00000031
IO$_SENSECHAR           =  0000001B
IO$_SENSEMODE           =  00000027
IO$_SETCHAR             =  0000001A
IO$_SETMODE             =  00000023
IO$_VIRTUAL             =  0000003F
IO$_WRITELBLK           =  00000020
IO$_WRITEPBLK           =  0000000B
IO$_WRITEVBLK           =  00000030
IOC$ALOUBAMAP              ********    X  03
```

```
IOC$LOADUBAMAPA                  ********  X   03
IOC$LOADUBAMAPN                  ********  X   03
IOC$MNTVER                       ********  X   03
IOC$RELMAPREG                    ********  X   03
IOC$REQCOM                       ********  X   03
IOC$REQMAPREG                    ********  X   03
IOC$RETURN                       ********  X   03
IOC$WFIKPCH                      ********  X   03
IO_DONE                          00000C70  R   03
IPC$_TIMER                     =  00000008
IRP$B_TYPE                     =  0000000A
IRP$C_LENGTH                   =  000000C4
IRP$L_ARB                      =  00000058
IRP$L_DIAGBUF                  =  0000004C
IRP$L_IOST1                    =  00000038
IRP$L_IOST2                    =  0000003C
IRP$L_MEDIA                    =  00000038
IRP$L_PID                      =  0000000C
IRP$L_SVAPTE                   =  0000002C
IRP$L_UCB                      =  0000001C
IRP$V_DIAGBUF                  =  00000007
IRP$V_FUNC                     =  00000001
IRP$W_BCNT                     =  00000032
IRP$W_BOFF                     =  00000030
IRP$W_CHAN                     =  00000028
IRP$W_FUNC                     =  00000020
IRP$W_STS                      =  0000002A
JIB$L_BYTCNT                   =  00000020
JIB$L_BYTLM                    =  00000024
LOAD_PORT                        00000987  R   03
LOAD_PORT_ALT                    0000099A  R   03
LOAD_PORT_AVAIL                  00000A16  R   03
LS_UCODE                       =  000003C0
MASKH                          =  00000080
MASKL                          =  08000000
MAX_C_BUFSIZE                  =  00003FFF
MAX_RCV                        =  00000007
MAX_XMT                        =  00000007
MMG$GL_SPTBASE                   ********  X   03
MOD$M_XM_BSEL1                 =  00000080
MOD$M_XM_HIGH                  =  00000001
MOD$M_XM_RS232                 =  00000010
MOD$M_XM_RS422                 =  00000020
MOD$V_XM_HIGH                  =  00000000
MOD$V_XM_INTMOD                =  00000002
MOD$V_XM_RS232                 =  00000004
MSG$_XM_ATTN                     ********  X   03
MSG$_XM_DATAVL                   ********  X   03
MSG$_XM_SHUTDN                   ********  X   03
NMA$C_CTCIR_BRC                =  000003E8
NMA$C_CTCIR_BSN                =  000003E9
NMA$C_CTCIR_DBR                =  000003F2
NMA$C_CTCIR_DBS                =  000003F3
NMA$C_CTCIR_DEI                =  000003FC
NMA$C_CTCIR_DEO                =  000003FD
NMA$C_CTCIR_LBE                =  00000411
NMA$C_CTCIR_LRT                =  00000407
NMA$C_CTCIR_RBE                =  00000410
NMA$C_CTCIR_RRT                =  00000406
NMA$M_CNT_COU                  =  00008000
NMA$M_CNT_MAP                  =  00001000
NMA$M_CNT_TYP                  =  00000FFF
NMA$V_CNT_MAP                  =  0000000C
NMA$V_CNT_UID                  =  0000000D
P1                             =  00000000
P2                             =  00000004
P3                             =  00000008
PCB$L_JIB                      =  00000080
PCB$L_PID                      =  00000060
POKE_USER                        00000CBA  R   03
PORT_INTR                        00000A71  R   03
PR$_IPL                        =  00000012
RCVFDT                           000001AA  R   03
RCV_B_BLKTYPE                    0000000A
RCV_B_MAPSLOT                    00000008
RCV_L_BACC                       0000000C
RCV_L_LINK                       00000000
RCV_START                        000001E5  R   03
RCV_T_DATA                       00000048
RCV_W_BLKSIZE                    00000008
REGDUMP                          00000C97  R   03
SCH$GL_PCBVEC                    ********  X   03
SCHED_FORK                       00000B5B  R   03
SENSEMODEFDT                     000003A6  R   03
SETMODEFDT                       00000231  R   03
SETMODEFDT_LINE                  00000321  R   03
SHUTDOWN                         00000DA5  R   03
SHUTDOWN_ALT                     00000DA7  R   03
SHUT_TIME                      =  000F4240
SIZ...                         =  00000001
SS$_ABORT                      =  0000002C
SS$_ACCVIO                     =  0000000C
SS$_BADPARAM                   =  00000014
SS$_CTRLERR                    =  00000054
SS$_DEVACTIVE                  =  000002C4
SS$_DEVOFFLINE                 =  00000084
SS$_ENDOFFILE                  =  00000870
SS$_INSFMAPREG                 =  00000344
SS$_NORMAL                     =  00000001
STARTIO                          0000045E  R   03
STARTUP                          00004C0   R   03
START_COMPLETE                   00000856  R   03
START_CTRL_ERROR                 00000849  R   03
START_ERROR                      0000084E  R   03
START_RECEIVE                    0000092B  R   03
START_REQ_PORT                   00000864  R   03
START_WAIT_PORT                  0000086A  R   03
TIMEOUT                          00000CFE  R   03
TQE$B_RQTYPE                   =  0000000B
TQE$B_TYPE                     =  0000000A
TQE$C_LENGTH                   =  00000030
TQE$C_SSSNGL                   =  00000001
TQE$L_FPC                      =  0000000C
TQE$L_FR3                      =  00000010
```

```
UCB$B_DEVCLASS          = 00000040
UCB$B_DEVTYPE           = 00000041
UCB$B_DIPL              = 0000005E
UCB$B_FIPL              = 0000000B
UCB$B_XM_DCER             00000132
UCB$B_XM_DCES             00000135
UCB$B_XM_DEVCNT           00000130
UCB$B_XM_FKB              00000138
UCB$B_XM_HCER             00000131
UCB$B_XM_HCES             00000134
UCB$B_XM_NBFR             00000130
UCB$B_XM_NBFS             00000133
UCB$B_XM_RCV_MAP          00000108
UCB$B_XM_RCV_MAX          0000010A
UCB$B_XM_REPR             00000137
UCB$B_XM_REPS             00000136
UCB$B_XM_XMT_MAP          00000109
UCB$B_XM_XMT_MAX          0000010B
UCB$C_LENGTH            = 00000090
UCB$C_XM_DEVCNT         = 00000008
UCB$C_XM_DRVCNT         = 00000004
UCB$C_XM_LENGTH           00000152
UCB$C_XM_QUEUES         = 00000008
UCB$L_AMB               = 00000060
UCB$L_CRB               = 00000024
UCB$L_DEVCHAR           = 00000038
UCB$L_DEVDEPEND         = 00000044
UCB$L_DUETIM            = 0000006C
UCB$L_FPC               = 0000000C
UCB$L_IRP               = 00000058
UCB$L_RCVBYTCNT           00000120
UCB$L_RCVMSGCNT           00000128
UCB$L_SVAPTE            = 00000078
UCB$L_XMTBYTCNT           00000124
UCB$L_XMTMSGCNT           0000012C
UCB$L_XM_AST              00000114
UCB$L_XM_BASEMAP          0000011C
UCB$L_XM_BASETAB          00000118
UCB$L_XM_DRVCNT           00000120
UCB$L_XM_LSTCSR           0000014C
UCB$L_XM_LSTPRT           00000148
UCB$L_XM_PID              00000110
UCB$L_XM_RCV_MAP          000000D0
UCB$L_XM_XMT_MAP          000000EC
UCB$M_INT               = 00000002
UCB$M_ONLINE            = 00000010
UCB$M_POWER             = 00000020
UCB$M_TIM               = 00000001
UCB$M_TIMOUT            = 00000040
UCB$M_XM_FORK_PEND      = 00002000
UCB$M_XM_FORK_XMT       = 00000001
UCB$M_XM_INITED         = 00000008
UCB$M_XM_LOSTERR        = 00001000
UCB$M_XM_NOTIF          = 00000800
UCB$Q_XM_PORT             000000A0
UCB$Q_XM_POST             000000B8
UCB$Q_XM_QUEUES           00000090
UCB$Q_XM_RCV_BUF          000000C0
UCB$Q_XM_RCV_MSG          000000C8
UCB$Q_XM_RCV_PND          000000B0
UCB$Q_XM_RCV_REQ          00000098
UCB$Q_XM_XMT_PND          000000A8
UCB$Q_XM_XMT_REQ          00000090
UCB$V_ONLINE            = 00000004
UCB$V_POWER             = 00000005
UCB$V_XM_FORK_PEND      = 0000000D
UCB$V_XM_FORK_XMT       = 00000000
UCB$V_XM_INITED         = 00000003
UCB$V_XM_LOSTERR        = 0000000C
UCB$V_XM_NOTIF          = 0000000B
UCB$W_BCNT              = 0000007E
UCB$W_BOFF              = 0000007C
UCB$W_DEVBUFSIZ         = 00000042
UCB$W_DEVSTS            = 00000068
UCB$W_ERRCNT            = 00000082
UCB$W_REFC              = 0000005C
UCB$W_STS               = 00000064
UCB$W_XM_MODSIG           00000150
UCB$W_XM_QUOTA            0000010C
UINST_CNF               = 00002296
UINST_RROM              = 0000814D
UNIT_INIT                 000000A4 R      03
VA$M_BYTE               = 000001FF
VA$S_VPN                = 00000015
VA$V_VPN                = 00000009
VEC$B_DATAPATH          = 00000013
VEC$B_NUMREG            = 00000012
VEC$L_IDB               = 00000008
VEC$L_UNITINIT          = 00000018
VEC$W_MAPREG            = 00000010
XM$DDT                    00000000 RG     03
XM$M_CHR_HDPLX          = 00000004
XM$M_CHR_MBX            = 00000010
XM$M_CHR_MOP            = 00000001
XM$M_CHR_SLAVE          = 00000008
XM$M_ERR_FATAL          = 00010000
XM$M_STS_ACTIVE         = 00000800
XM$V_CHR_LOOPB          = 00000001
XM$V_CHR_MBX            = 00000004
XM$V_STS_ACTIVE         = 0000000B
XM$V_STS_BUFFAIL        = 0000000C
XMTFDT                    000000D0 R      03
XMT_START                 0000010E R      03
XMT_START_ALT             00000113 R      03
XM_END                    00000F36 R      03
XM_E_M_LOST             = 00000010
XM_E_M_MOP              = 00000008
XM_E_M_NONEXMEM         = 00000100
XM_E_M_POWER            = 00000400
XM_E_M_PROCERR          = 00000200
XM_E_M_START            = 00000080
XM_E_M_TIMEOUT          = 00000800
XM_E_V_POWER            = 0000000A
XM_E_V_PROCERR          = 00000009
```

```
XM_E_V_TIMEOUT                  = 0000000B
XM_I_CSR                          00000000
XM_I_M_IEI                      = 00000040
XM_I_M_LOOPB                    = 00000800
XM_I_M_MCLR                     = 00004000
XM_I_M_RCV                      = 00000004
XM_I_M_RDI                      = 00000080
XM_I_M_ROMI                     = 00000200
XM_I_M_ROMO                     = 00000400
XM_I_M_RQI                      = 00000020
XM_I_M_RUN                      = 00008000
XM_I_M_STEPUP                   = 00000100
XM_I_V_RQI                      = 00000005
XM_O_CSR                          00000002
XM_O_M_IEO                      = 00000040
XM_O_M_RDO                      = 00000080
XM_O_V_RCV                      = 00000002
XM_O_V_TYPE                     = 00000000
XM_PORT                           00000004
XM_UCODE                          00000006
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000152 (338.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	00000069 (105.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00000F36 (3894.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.04	00:00:01.11
Command processing	120	00:00:00.44	00:00:05.02
Pass 1	828	00:00:26.80	00:01:32.21
Symbol table sort	0	00:00:03.68	00:00:12.70
Pass 2	505	00:00:06.16	00:00:24.08
Symbol table output	1	00:00:00.23	00:00:02.57
Psect synopsis output	0	00:00:00.04	00:00:00.51
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1485	00:00:37.40	00:02:18.21

The working set limit was 2400 pages.
223463 bytes (437 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3446 non-local and 183 local symbols.
2525 source lines were read in Pass 1, producing 27 object records in Pass 2.
59 pages of virtual memory were used to define 55 macros.

```
                          +----------------------------+
                          ! Macro library statistics !
                          +----------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SHRLIB]NMALIBRY.MLB;1                   1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                      34
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   11
TOTALS (all libraries)                              46

3627 GETS were required to define 46 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:XMDRIVER/OBJ=OBJ$:XMDRIVER MSRC$:XMDRIVER/UPDATE=(ENH$:XMDRIVER)+EXECML$/LIB+SHRLIB$:NMALIBRY/LIB
```

0121 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

XQDRIVER
LIS

XMDRIVER
LIS